FACULTY OF
COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA

Perpustakaan SKTM

IMPLEMENTING TRAFFIC CONDITIONING USING TOKEN BUCKET AND LEAKY BUCKET

Submitted by
CHIA KAI YAN (WEK010044)

Under the Supervision of
MR. PHANG KEAT KEONG

Moderator
MR. ANG TAN FONG

SESSION 2003/2004

University of Malaya

ABSTRACT

Since network is expanding from time to time, therefore network simulator is required as a tool investigate performance of the network effectively. Without building a real network, the performance of the network can be analyzed. UMJaNetSim is a network simulator that provides graphical user interface for user to allocate the components and set the respective parameters virtually and get a prediction result.

UMJaNetSim is running in ATM environment recently. Based on the result of research, the performance of network simulator will be increase if UMJaNetSim run in Diffserv environment. In order to develop UMJaNetSim in a Diffserv environment, Traffic Conditioner is required. This project focuses on the development for Traffic Conditioning using Token Bucket and Leaky Bucket. By implementing a Traffic Conditioner in UMJaNetSim, a more effective traffic flow handling system which consists of metering, marking, shaping and dropping stages will be produced. Besides that, Traffic Conditioner will avoid the traffic flow from any traffic congestion.

Furthermore, the simulator is capable of multithread operation and platform independent. Thus, the developed simulator can run on different platforms like Windows or UNIX.

University of Malaya

ACKNOWLEDGEMENT

I would like to take this opportunity to dedicate my sincere appreciation to those people who had helped me throughout this project. His precious advises and sound understanding throughout the project is deeply appreciated.

First of all, I would like to express my gratitude to my respected supervisor, Mr. Phang Keat Keong for his support, valuable guidance, encouragement and constructive comments during this project.

Secondly, special thanks to Mr. Ang Tan Fong for being a considerate and kind moderator who has contributed suggestions and ideas in this project. Besides that, I would like to express my appreciation to Mr. Ling Teck Chaw for his brilliant opinions.

Last but not least, I will like to thanks all my group members for this project, Andrew Chiam Ming Jer, Lim Lee Wen, Tang Geck Hiang, Chee Wai Hong, Chan Chin We, Malini and Au Yee Boon for their generous sharing of knowledge and support. These supports contribute to the completion of my project.

University of Malaya

TABLE OF CONTENT

University of Malaya

University of Malaya

University of Malaya

University of Malaya

University of Malaya

LIST OF FIGURES

University of Malaya

University of Malaya

LIST OF TABLES

University of Malaya

CHAPTER 1: INTRODUCTION

1.1 Introduction to Network Simulator

With the rapid development of high-speed network, network simulator has become a valuable tool to study and investigate the protocol and design issues regarding the performance of the network. A network simulator provides a means for researches and network planners to analyze the behavior of the network without the expense of building a real network. The simulator is a tool that gives the user an interactive modeling environment with a graphical user interface. With this tool the user may create different network topologies, control component parameters, measure network activity and log data from simulation runs.

As a planning tool, a network planner can run the simulator with various network configurations and traffic loads to obtain statistics such as utilization of network links and throughput rates of virtual circuits. It could be used to answer questions such as where will be the bottlenecks in the planned network or what is the effect of changing the speed of a link and will adding a new application cause congestion. Statistics are reported directly to the screen or logged in a data file for further processing. As a protocol analysis tool, a researcher or protocol designer could study the total system effect of a particular protocol.

University of Malaya

1.2 Introduction to Traffic Conditioning

Traditional IP network can only provide all customers with Best-Effort (BE) services where Best-Effort service means all the in packets will send into traffic flow without classification. Subsequently, all the traffic competes equally for network resources. With the development of new applications of Internet, such as voice, video and www, the desire of QoS becomes more and more strong. Recently IETF proposes the Differentiated Services architecture to provide scalable means to deliver IP QoS based on handling of traffic aggregates. Traffic Classification state is conveyed by means of IP-layer packet marking using the DS field. Packets are classified and marked to receive a particular PHB (per-hop behavior) on nodes along their path. Sophisticated classification and traffic conditioning, including marking, policing, and shaping operations, need only be implemented at network boundaries.

1.3 Project Objective

The main objectives of this project are to study and understand the existing UMJaNetSim and traffic conditioning techniques commonly used in network simulator. Apply the knowledge gained through research then develop traffic conditioning in a Differentiated Service environment into UMJaNetSim. By applying the traffic conditioning in UMJaNetSim, Quality of Service (QoS) of the system will be increased and flow congestion will be avoided.

University of Malaya

1.4 Goals of Project

The main goal of the project is to implement traffic conditioning into UMJaNetSim by using Token Bucket and Leaky Bucket. The following chapter will explain the concepts of token bucket and leaky bucket algorithm that will be applied into UMJaNetSim in order to develop a traffic conditioner to handle traffic flow. The expected outcomes of the project are a less congested traffic flow and a more efficient traffic handling system.

1.5 Project Scope

In order to successful applying traffic conditioning into UMJaNetSim, I had chosen some effective methods which are:

- Joining the discussion with my project group and Mr. Phang every Friday. Each of us had shared our knowledge in respective area and explains to others by power point slide during discussion time.
- Doing researches on Internet. I used some search engine like www.google.com and www.yahoo.com to search for sufficient information which is related to Network Simulator.
- Refer to senior project for some idea. I had borrow some project from senior where the title are related to network simulator and gain some idea on how to implement traffic conditioning in UMJaNetSim.

University of Malaya

1.6 Project Schedule

ID	Task Name	Duration	Start	Finish
1	Project Definition	10 days	Mon 6/16/03	Fri 6/27/03
2	Literature Review	22 days	Fri 6/27/03	Mon 7/28/03
3	Research Works	20 days	Mon 7/21/03	Fri 8/15/03
4	System Analysis	15 days	Mon 8/11/03	Fri 8/29/03
5	System Design	20 days	Wed 8/20/03	Tue 9/16/03
6	Implementation	80 days	Wed 9/17/03	Tue 1/6/04
7	Testing	23 days	Fri 1/2/04	Tue 2/3/04
8	Documentation	160 days	Mon 6/30/03	Fri 2/6/04

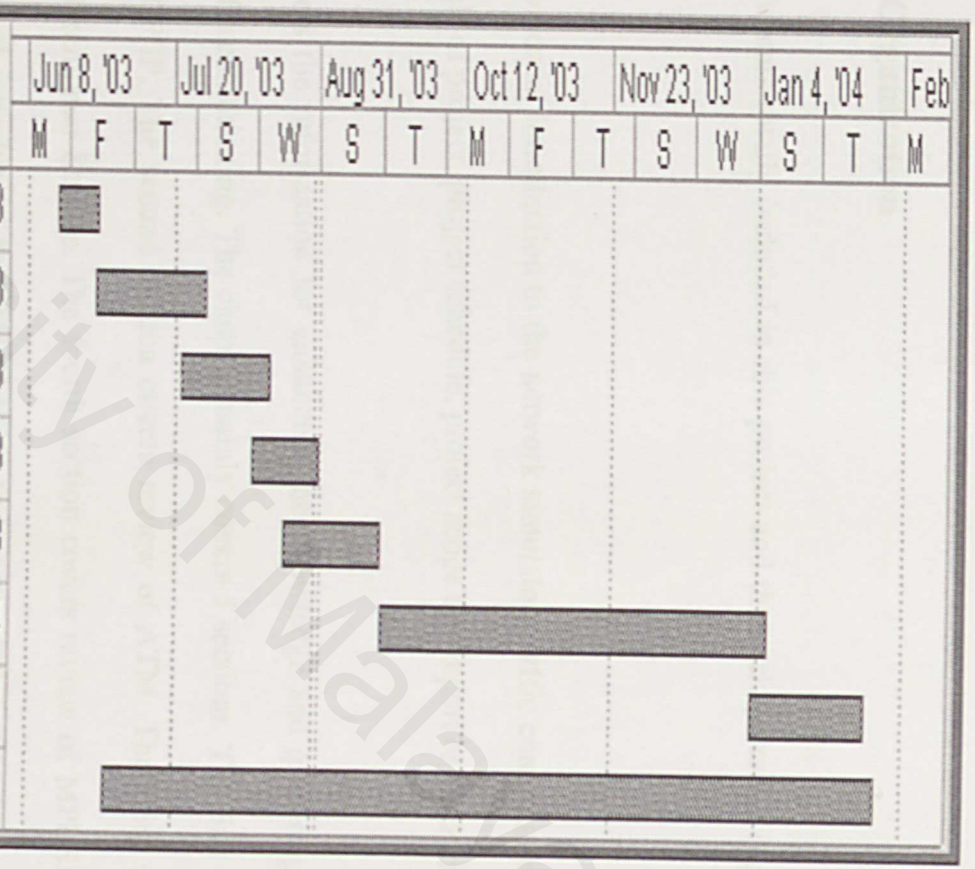


Figure 1.1: Project Schedule Timeline

University of Malaya

1.7 Report Organization

There are 5 chapters totally included in this project and those chapters are organized as follow:

Chapter 1 provides an introduction to the network simulator, traffic conditioning, project objectives, goals of project, project schedule, project scope and report organization.

Chapter 2 covers the information for enhancing the knowledge and gathering of basic concept in traffic conditioning. The chapter mainly covers 5 sections. The first section is a review of TCP/IP. The second section covers review of ATM. The third section is explaining Differentiated Service. The fourth section covers review of MPLS. The last section consists of introducing to UMJaNetSim and comparing few existing network simulator.

Chapter 3 explains how to implement traffic conditioning in a Differentiated Service environment. Besides that, explanations of token bucket and leaky bucket algorithm by example will be included. In this section, how the system meter, shape and drop the packets by following the parameters that had been preset by the user and policies that preset by administrator.

University of Malaya

Chapter 4 discusses the platform, programming language and development tools chosen to create the network simulator components. Each function for the main classes in UMJaNetSim will be explained.

Chapter 5 includes the design for the Traffic Conditioner and describing how will it be implemented into coding. Besides that, some flow chart for handling the traffic conditioning by token bucket and leaky bucket will be shown in this section.

Chapter 6 describes the major function of the methods or java files added into UMJaNetSim. Each function will be explained clearly here. In this section, the coding will be shown be explained.

Chapter 7 shows the unit, class and system testing for the Traffic Conditioner including Single Leaky Bucket, Double Leaky Bucket and Token Bucket. The result for testing stage can proved that the Traffic Conditioner is successfully implemented.

Chapter 8 concludes the research and development of the network topology. And summarize the findings of the project, the final product and the constraint during the development and testing stage.

University of Malaya

CHAPTER 2: LITERATURE REVIEW

This chapter covers 5 sections that are information and basic concepts to implement a traffic conditioner in UMJaNetSim. The first four section explaining TCP/IP, Differentiated Service, ATM and MPLS. The last section consists of introducing UMJaNetSim and comparing few existing network simulator.

2.1 TCP/IP

Transmission Control Protocol/Internetworking Protocol (TCP/IP) is an industry standard suite of protocols providing communication in a heterogeneous environment. It provides a routable, enterprise networking protocol and access to the Internet and its resources. It is also a set of protocol to define how all transmissions are exchanged across the Internet. TCP/IP has been widely use because of its effectiveness. It was initially successful because it delivered a few basic services that everyone needs (file transfer, electronic mail, remote logon) across a very large number of client and server systems. (Forouzan, 2001)

TCP and IP were developed to connect a different network and different vendors into a network of Internet designed it. Several computers in a small department can use TCP/IP with other related protocols on a single LAN. The IP component provides routing from the department to the enterprise network then to regional networks and finally to the

University of Malaya

global Internet. In the real world of Internet, communication network will facing lost of information while process of transferring therefore the Department of Defense (DOD) research project designed TCP/IP to be robust and solve the problem by automatically recover from any failure. This design allows the construction of very large networks with less central management. TCP is a connection – oriented protocol where session is established before exchanging data.

2.1.1 TCP/IP components

TCP/IP is decomposed into several layers in order to works with other communications protocol that is

Internet Protocol - IP is responsible for moving packet of data from node to node. IP forwards each packet based on a four-byte destination address. The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world. IP is the transmission mechanism used by TCP/IP protocols. IP provides an unreliable and connectionless service that also called datagram service. It means that IP does not guarantee that transmitted packets will be delivered and each packet or datagram is handled independently. Since IP is not aware that packets between hosts may be sent in a logical sequence. Therefore using IP as the transmission mechanism will consequence for the lost of packets, duplicate packets and packets are delivered out-of-sequence. Packets in the IP

University of Malaya

layer called datagram. Figure 1.1 shows the IP datagram format for IP version 4 (IPv4).

a) 20 bytes <= Header Size <= 24 * 32 bit-words = 60 bytes

b) 20 bytes <= Total Length <= 216 bytes = 65536 bytes

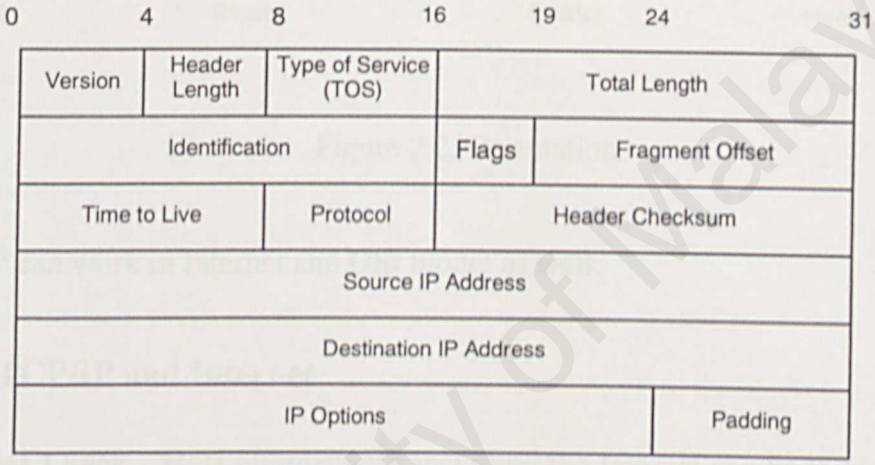


Figure 2.1: IPv4 Datagram

TCP - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received therefore it is called as a connection-oriented protocol that provides a reliable unicast end-to-end byte stream over an unreliable internetwork.

Sockets - is a name given to the package of subroutines that provide access to TCP/IP on most systems.

University of Malaya

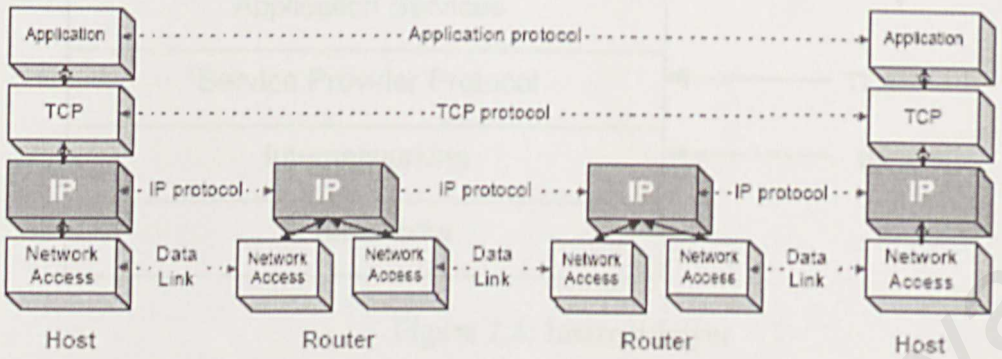


Figure 2.2: Orientation

TCI/IP can work in Internet and OSI model as well:

2.1.2 TCP/IP and Internet

Internet Layer - Most internetwork including the Internet can be thought of as a layered architecture to simplify understanding. The layer concept helps in developing applications for internetwork too. The layering also shows how the different parts of TCP/IP work together. These layers as conceptual only and they are not really physical or software layers as such OSI or TCP/IP layers. There are some references mentions that Internet layer divided into five layers, but it is convenient to think of the Internet as having four layers. This layered Internet Layer Architecture is shown in Figure 2.3.

University of Malaya

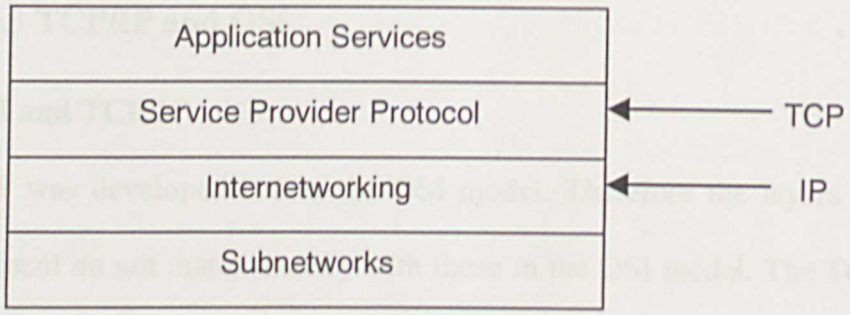


Figure 2.3: Internet Layer

a) Internetwork layer that is on top of the subnetwork layer is providing the function for networks through gateways to interact. Each subnetwork uses gateways to connect to the other subnetwork in the internetwork. Data in internetwork layer transferred from gateway to gateway until it reaches its destination and then passes into the subnetwork layer. The internetwork layer runs the IP.

b) The Service Provider Protocol layer is responsible for the overall end-to-end communications of the network. This is the layer runs the TCP and other protocols. It handles the data traffic flow itself and ensures reliability for the transformations of messages.

c) The top layer is the Application Services layer which supports the interfaces to the user applications. This layer interfaces to electronic mail, remote file transfers, and remote access. There are several protocols used in this layer.

University of Malaya

2.1.3 TCP/IP and OSI

OSI and TCP/IP

TCP was developed before the OSI model. Therefore the layers in the TCP/IP protocol do not match exactly with those in the OSI model. The TCP/IP protocol is made of four layers. There are subnetwork, internetworking, service provide protocol, application services. Figure 2.4 shows relation between TCP/IP and OSI layer.

The application layer in TCP/IP can be equated with the combination of session, presentation, and application layers of the OSI model. At the service provider protocol layer, TCP/IP defines two protocols that are TCP and User Datagram Protocol (UDP). At the internetworking layer, the main protocol defined by TCP/IP is Internetworking Protocol (IP) although there are some other protocols that support data movement in this layer. At the subnetwork layer, TCP/IP does not define any specific protocol. A network in a TCP/IP internetwork can be a Local Area Network (LAN), a Metropolitan Area Network (MAN) or Wide Area Network (MAN)

OSI Model Upper Layers

OSI combine the application, presentation and session layers as upper layers. Generally, these layers perform application-specific functions like data

University of Malaya

formatting, encryption and connection management. Examples of upper layer technologies in the OSI model are HTTP, SSL and NFS.

OSI Model Lower Layers

The remaining lower layers provide more primitive network-specific functions like routing, addressing and flow control. Examples of lower layer technologies in the OSI model are TCP, IP, and Ethernet.

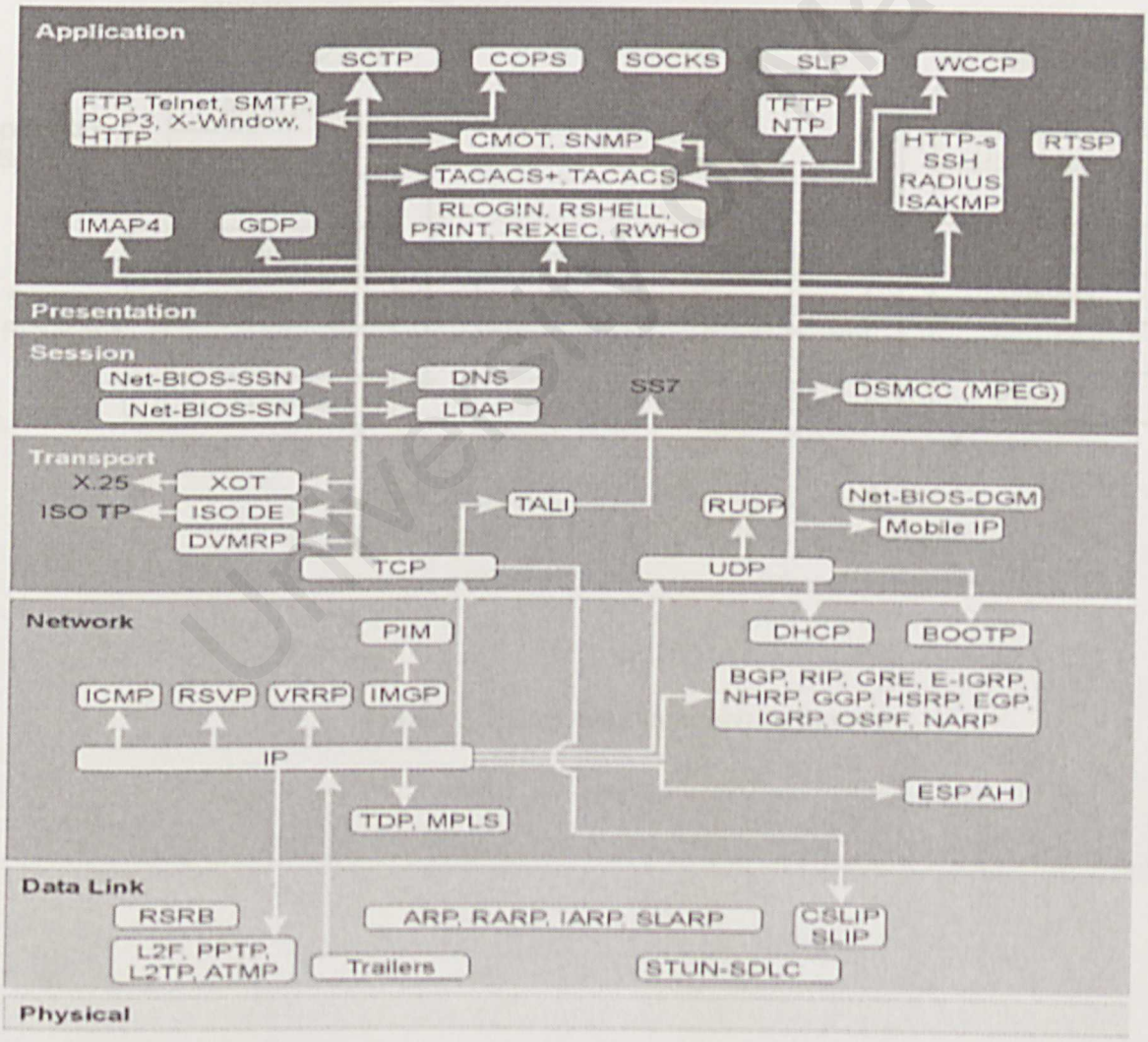


Figure 2.4: Relation between OSI and TCP/IP layer

A simple example for how the Internet Architecture layer model works shown in Figure 2.5. Assume an application on one machine wants to transfer a datagram to an application on another machine in a different subnetwork. The layers in the sending and receiving machines are the OSI layers, with the equivalent Internet Architecture layers indicated.

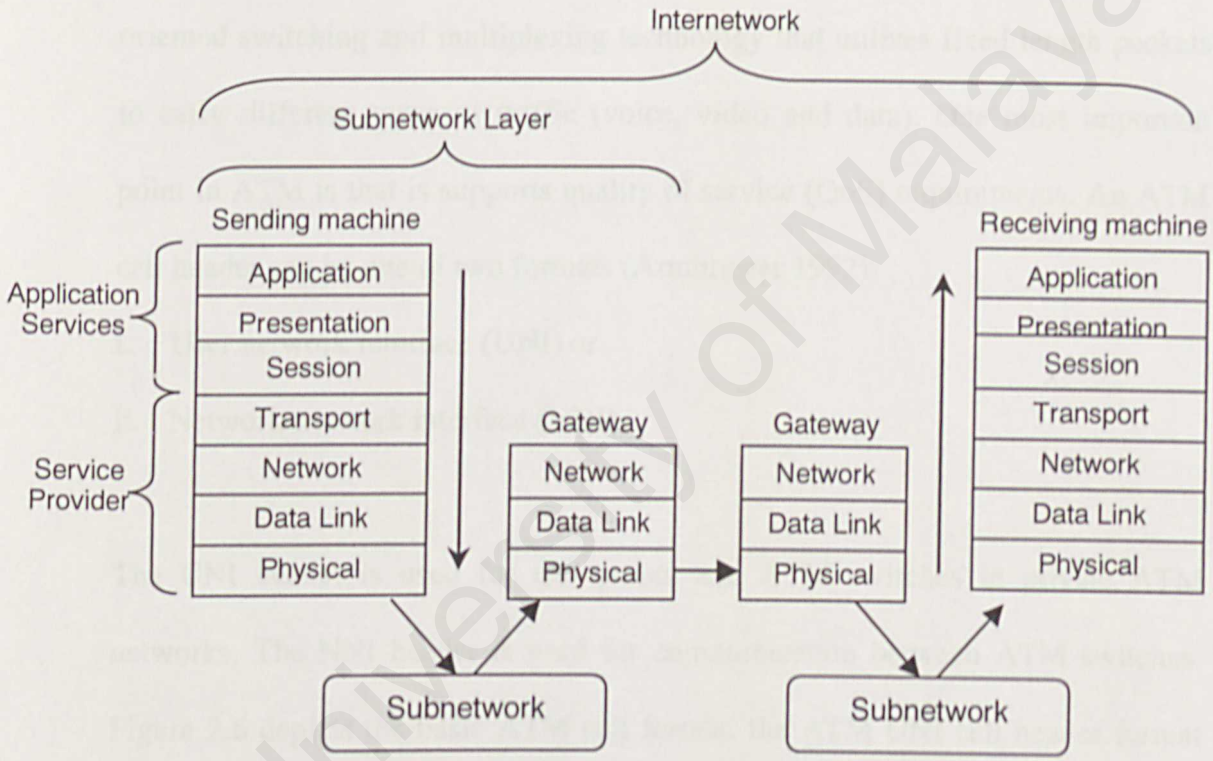


Figure 2.5: Internet Architecture Layer

University of Malaya

2.2 ATM

2.2.1 Introduction to Asynchronous Transfer Mode (ATM)

Asynchronous Transfer Mode (ATM) is the primary networking technology for next generation, multimedia communication. ATM is a high performance, cell oriented switching and multiplexing technology that utilizes fixed length packets to carry different types of traffic (voice, video and data). One most important point in ATM is that is supports quality of service (QoS) requirements. An ATM cell header can be one of two formats (Armbruster 1992):

i. User network interface (UNI) or

ii. Network-network interface (NNI)

The UNI header is used for end points and ATM switches in private ATM networks. The NNI header is used for communication between ATM switches. Figure 2.6 depicts the basic ATM cell format, the ATM UNI cell header format and the ATM NNI cell header format.

ATM transfers information in fixed size units called cells. Each cell consists of 53 octets or bytes. The first 5 bytes contain cell header information and the remaining 48 contain the payload (user information). Small fixed length are well suited to transfer voice and video traffic because such traffic is intolerant of delays that result from having to wait for a large data packet to download.

University of Malaya

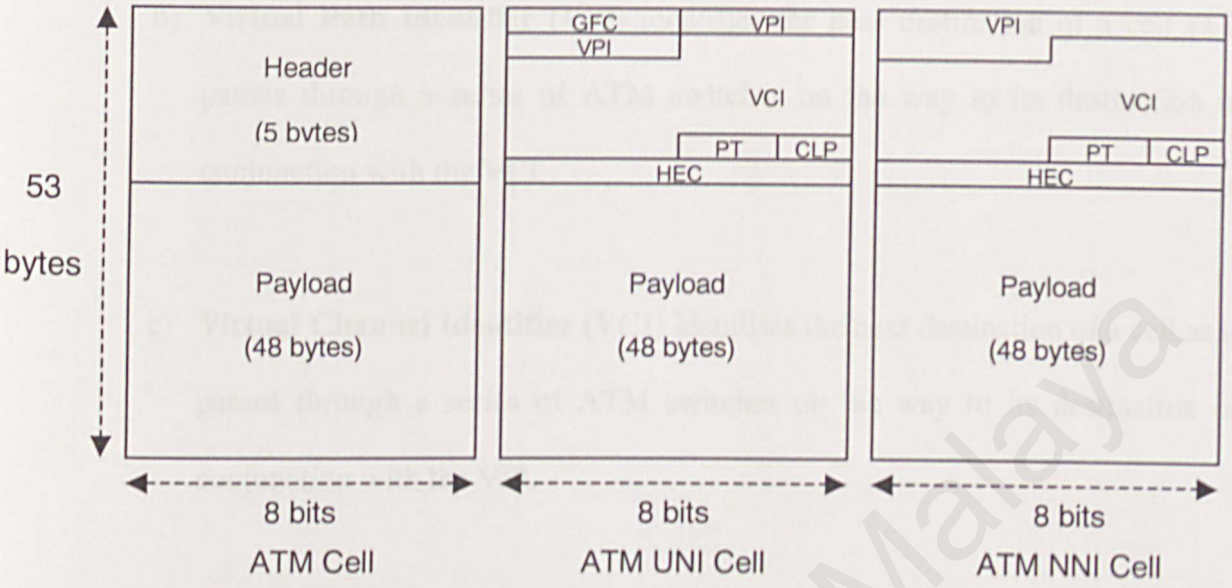


Figure 2.6: An ATM cell, UNI cell and ATM NNI cell header

Unlike to UNI, the NNI header does not include the Generic Flow Control (GFC) field. Additionally, the NNI header has a Virtual Path Identifier (VPI) field that occupies the first 12 bits allowing for larger trunks between public ATM switches. In additional to GFC and VPI header fields, several others are used in ATM cell header fields. The following descriptions summarize ATM cell header fields illustrated in Figure 2.6.

a) **Generic Flow Control (GFC)** provides local functions such as identifying multiple stations that share a single ATM interface. This field is typically not used and is set to its default value.

University of Malaya

b) **Virtual Path Identifier (VPI)** identifies the next destination of a cell as it passes through a series of ATM switches on the way to its destination in conjunction with the VCI.

c) **Virtual Channel Identifier (VCI)** identifies the next destination of a cell as it passes through a series of ATM switches on the way to its destination in conjunction with the VPI.

d) **Payload Type (PT)** indicates in the first bit whether cell contains user data or control data. If the cell contains user data, the second bit indicates congestion and the third bit indicates whether the cell is the last in a series of cells that represent a single AAL5 frame.

e) **Congestion Loss Priority (CLP)** indicates whether the cell should be discarded if encounters extreme congestion as it moves through the network. If the CLP bit equals 1, the cell should be discarded in preference to cells with the CLP bit equal to zero.

f) **Header Error Control (HEC)** calculates checksum only on the header itself.

University of Malaya

2.2.2 Introduction to ATM service classes

It is very complex to provide desired QoS for different applications. For example, voice is delay sensitive but not loss sensitive, data is loss sensitive but not delay sensitive while some other applications may be both delay sensitive and loss sensitive.

To make it easier to manage, the traffic in ATM is divided into the following five service classes:

i. Constant Bit Rate (CBR)
ii. Real Time Variable Bit Rate (rt-VBR)
iii. Non Real Time Variable Bit Rate (nrt-VBR)
iv. Unspecified Bit Rate (UBR)
v. Available Bit Rate (ABR)

These service categories relate traffic characteristics and QoS requirement to network behavior. Functions such as routing, CAC and resource allocation are in general structured differently for each service category. Service categories are distinguished as being either real time or non real time.

University of Malaya

For real time traffic, there are two categories:

a) **CBR** is used by connections that request a static amount of bandwidth that is continuously available during the connection lifetime.

b) **rt-VBR** is intended for real time applications such as those requiring tightly constrained delay variation as would be appropriate for voice and video application.

For Non real time traffic there are three categories:

a) **nrt-VBR** is intended for non real time applications and this service may support statistical multiplexing of connections. No delay bounds are associated with this service category such as multimedia e-mail.

b) **UBR** is intended for non real time applications such as those requiring tightly constrained delay and delay variation.

c) **ABR** is an ATM layer service category for which the limiting ATM layer transfer characteristics provided by the network may change subsequent to connection establishment.

University of Malaya

2.3 Differentiated Service

2.3.1 Integrated Services

By emerging multimedia services on the Internet, the best-effort-only service architecture turns out to be fairly inconsistent with the requirement of the new types of applications therefore a service model called Integrated Services (IntServ) to cope with the demands for the Quality of Service (QoS). The IntServ framework was setup to allow certain types of applications to select QoS level between multiple levels of delivery service classes. IETF defines two other service classes besides best-effort service:

i)Controlled load service

Controlled load service is designed to assure users to get a service that closely approximate best-effort effort within an unloaded or lightly utilized network. However, it does not provide any firm quantitative guarantees.

ii)Guaranteed service

The Guarantees QoS service is intended for delay-sensitive applications. It gives the users an assured amount of bandwidth, firm end-to-end delay bounds and no packet loss when conforming to the parameters negotiated at the connection setup period.

University of Malaya

2.3.2 Differentiated service

As the Internet evolves into a global commercial infrastructure, there is a growing need to support more enhanced services than IntServ. Differentiated services (Diffserv) are a scalable solution to Quality of Service (QoS) while there are many users and high usage of bandwidth. Diffserv was introduced since internet requires keeping a great deal of state information because of its flow-based nature but the IntServ model could not scale well as the number of user and the span of the network keep growing. (Ion 2003)

The Diffserv mechanism also allows providers to allocate different levels of service to different users of the Internet to support various types of applications. It is a relatively simple mechanism that does not depend entirely on per flow resource reservation. Instead, the DiffServ architecture is designed to offer service to aggregated service flow, where a number of individual session flows are grouped and treated consistently by the network. The grouping of the traffic is according to the marking of each packet at the boundary of the network. The marking is based on the Type of Service (TOS) field within the IPv4. Normally, this field is called DS field as shown in Figure 2.7. The routers within the network to indicate a particular Per-Hop-Behavior (PHB) that is a forwarding treatment interpret the value in the DS field.

University of Malaya

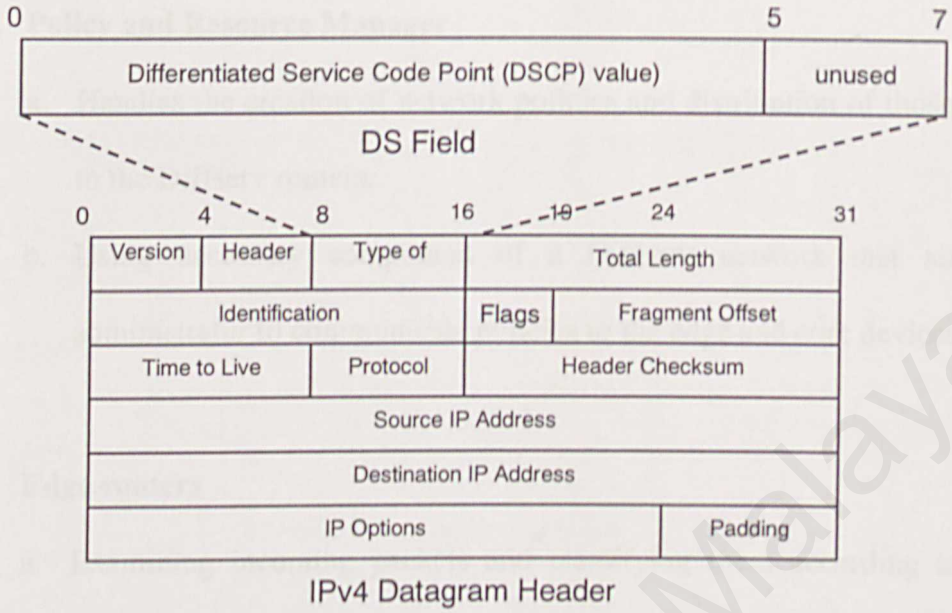


Figure 2. 7: DS field format from within TOS field in IPv4

Diffserv Architecture

The Diffserv architecture has three major components as shown in Figure 2.8 which are Policy and Resource Manager, Edge Router and Core Router

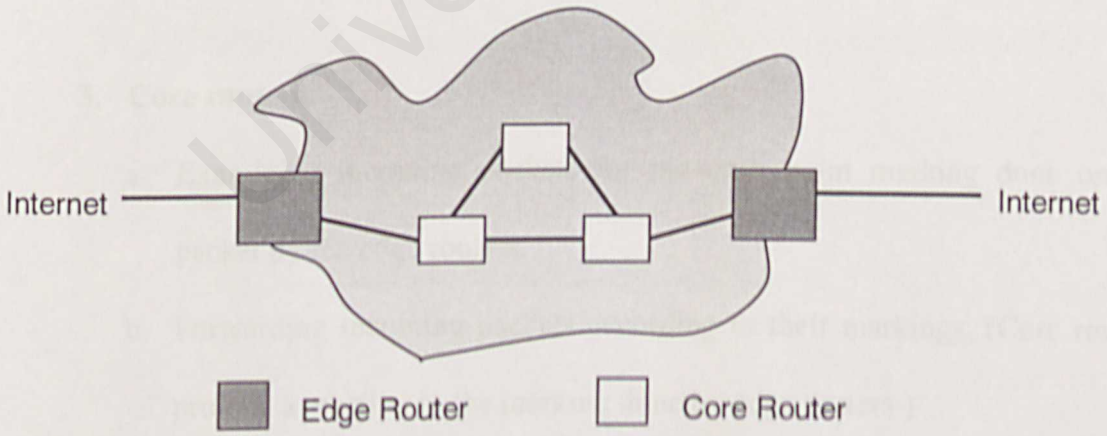


Figure 2.8: Devices in Diffserv Domain

University of Malaya

1. **Policy and Resource Manager**
   a. Handles the creation of network policies and distribution of those policies to the Diffserv routers.
   b. Using necessary component of a Diffserv network that allows an administrator to communicate policies to the edge and core devices

2. **Edge routers**
   a. Examining incoming packets and classifying them according to policy specified by the network administrator.
   b. Marking packets with a code point that reflects the desired level of service.
   c. Ensuring that user traffic adheres to its policy specifications, by shaping and policing traffic

3. **Core routers**
   a. Examining incoming packets for the code point marking done on the packet by the edge routers.
   b. Forwarding incoming packets according to their markings. (Core routers provide a reaction to the marking done by edge routers.)

University of Malaya

Traffic Conditioning in Diffserv

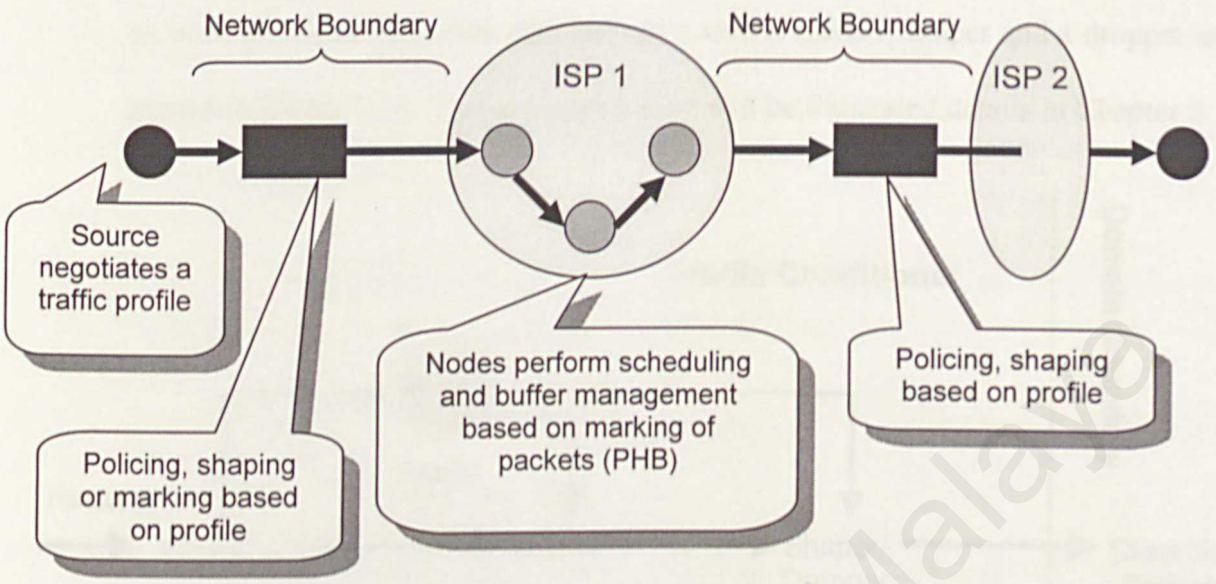


Figure 2.9: Traffic Conditioner function at Core and Edge Router

When entering a DiffServ network, each IP packet is classified as to their related priority before it is passed through an admission filter where traffic is shaped and conditioned to meet the policy requirements associated with the classification. The shaped traffic is then assigned to an aggregated traffic flows based on their different priority request as aforementioned by marking the IP header DS field of the component packets with the appropriate DSCP (Differentiated Services Code Point). The Per-Hop behaviors (PHB) of all the intermediate components of the DiffServ are then performed when forwarding the traffic flows in the interior network. (Jean 2003)

Generally, traffic conditioners at the edge contain various elements such as classifier, meter, marker, shaper and dropper. A traffic conditioner is referred to

as a set of components that may include a meter, marker, shaper and a dropper as shown in Figure 2.10. Traffic Conditioners will be illustrated details in Chapter 3.

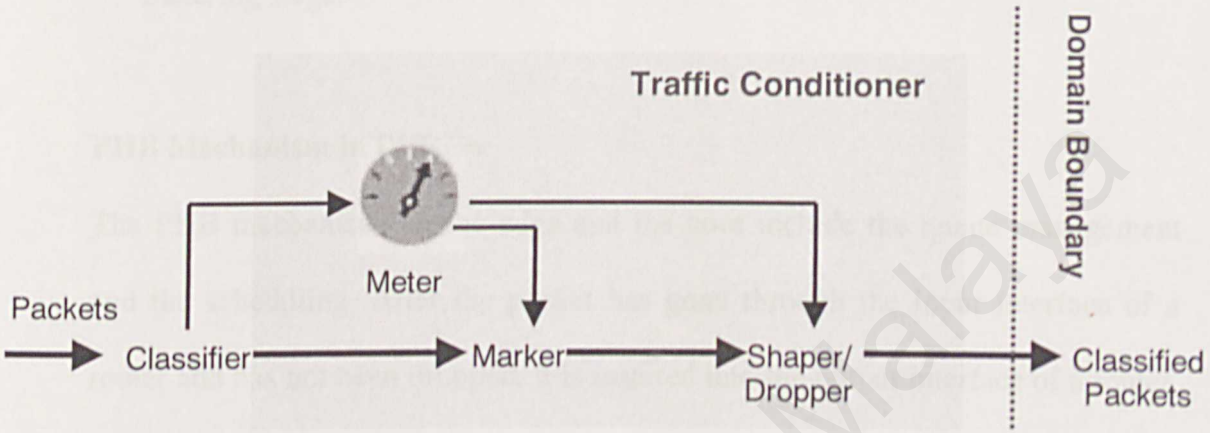


Figure 2.10: Traffic Conditioner includes meter, marker, shaper and dropper

- **Classifier**

  Select a packet in a traffic stream based on the content of some portion of the packet header.

- **Meter**

  Check compliance to traffic parameter and passes results to marker and shaper/dropper to trigger action for in/out-of-profile packets.

- **Marker**

  Writes / rewrites the DSCP value.

- **Shaper**

  Delay some packets for them to be compliant with the profile.

University of Malaya

- **Dropper**

  Discard the non-conforming packets which are violating the policy in metering stage.

PHB Mechanism in Diffserv

The PHB mechanisms at the edge and the core include the queue management and the scheduling. After the packet has gone through the input interface of a router and has not been dropped, it is inserted into the output interface of a router. This queue can be a simple queue holding all the traffic classes or a number of queues where each one holds one distinct traffic class. There are various queuing algorithms but only the ones implemented are described here. The PHB mechanisms include packet scheduling mechanism and queuing management. There are four types of mechanisms available that are:

- The **Default PHB** essentially specifies that a packet marked with a DSCP value as recommended of '000000' gets the traditional best effort service from a DS compliant node (a network node that complies to the entire core DiffServ requirements). Also, if a packet arrives at a DS-compliant node and its DSCP value is not mapped to any of the other PHBs where it will get mapped to the default PHB.
- **Class-Selector PHB** preserves backward compatibility with the IP Precedence scheme, DSCP values as recommended of 'xxx000' where x is

either 0 or 1 are defined. These code points are called Class-Selector code points. These PHBs ensure that DS compliant nodes can co-exist with IP Precedence aware nodes, with the exception of the DTS bits.

- **Expedited Forwarding (EF)** PHB is the key ingredient in DiffServ for providing a low-loss, low-latency, low-jitter, and assured bandwidth service. Although EF PHB when implemented in a DiffServ network provides a premium service, it should be specifically targeted toward the most critical applications, since if congestion exists, it is not possible to treat all or most traffic as high priority. EF PHB is especially suitable for applications (like VoIP) that require very low packet loss, guaranteed bandwidth, low delay, and low jitter. The recommended DSCP value for EF is '101110'.

- **Assured Forwarding (AFxy) PHB** – It defines a method by which Behavior Aggregates can be given different forwarding assurances following dropping precedence as the Table 2.1

Drop Precedence	Class 1	Class 2	Class 3	Class 4
Low Drop Precedence	(AF 11) 001010	(AF 21) 010010	(AF 31) 011010	(AF 41) 100010
Medium Drop Precedence	(AF 12) 001100	(AF 22) 010100	(AF 32) 011100	(AF 42) 100100
High Drop Precedence	(AF 13) 001110	(AF 23) 010110	(AF 33) 011110	(AF 43) 100110

Table 2.1: Diffserv AF Code Point table

University of Malaya

SLA

The core of the whole service is the Service Level Agreement (SLA), or policy. The SLA may specify packet classification and re-marking rules and may also specify traffic profiles and actions to traffic streams which are in or out-of-profile.

2.4 MPLS

2.4.1 Traditional Routing

In traditional routing environments, a packet is forwarded through a network on a hop-by-hop basis using interior gateway protocols (IGPs) such as routing information protocol (RIP) and open shortest path first (OSPF). The packet also can forward through exterior gateway protocol (EGPs) such as border gateway protocol (BGP). Referencing the destination Layer 3 addresses against a routing table for a next hop entry does this. To clarify, each router that a packet traverses must do a route lookup table based on that destination Layer 3 address in the IP header. This must be performed to determine the packet's next hop in its path to get it to the final destination. The Layer 2 destination address is then replaced with the address of the next hop's Layer 2 address, and the source Layer 2 address is then replaced with the Layer 2 address of the current router leaving the source and destination Layer 3 addresses in place for the next hop to perform its own

University of Malaya

route lookup table on the packet. This process must be repeated at each hop to deliver the packet to its final destination.

Traditional routing is no longer useful in a network are as network size is growing rapidly recently, therefore traditional routing will facing the problem of overhead and slow respond time since the routing lookup table have to update in a short interval time to confirm every router or node are function properly.

2.4.2 MPLS

In order to meet the growing demand for high usage of bandwidth, Internet service providers (ISPs) need higher performance switching and routing products. Although most carrier and service provider core networks run on impressive ATM backbones, most connections to these providers continue to be slow frame relay and point-to-point connections therefore there are latency and sometimes bottlenecks at the edge access points. Core network routers also contribute to latencies since each must make its own individual decision on the best way to forward each incoming packet. Traditionally, IP has been routed over ATM using IP over ATM via virtual circuits (VCs) or multi protocol over ATM (MPOA). These forwarding methods were inefficient and complicated. Therefore MPLS was introduced to meet the traffic management features and performance of

University of Malaya

traditional switches combined with the forwarding intelligence of a router because it integrates the key features of both Layer 2 and Layer 3.

Multi protocol label switching (MPLS) is a versatile solution to address the problems of speed, scalability, routing, quality-of-service (QoS) management and traffic engineering exist over existing asynchronous transfer mode (ATM) and frame-relay networks. MPLS enable the function of bandwidth-management and service requirements for IP based on backbone networks. MPLS performs the following functions:

a) Specifies mechanisms to manage traffic flows of various granularities, such as flows between different hardware, machines, or even flows between different applications
b) Remains independent of the Layer-2 and Layer-3 protocols
c) Map IP addresses to simple, fixed-length labels used by different packet-forwarding and packet-switching technologies
d) Interfaces to existing routing protocols such as resource reservation protocol (RSVP) and open shortest path first (OSPF)
e) Supports the IP, ATM and frame-relay Layer-2 protocols

University of Malaya

The MPLS architecture provides basic services by a server to few clients as shown at Figure 2.11.

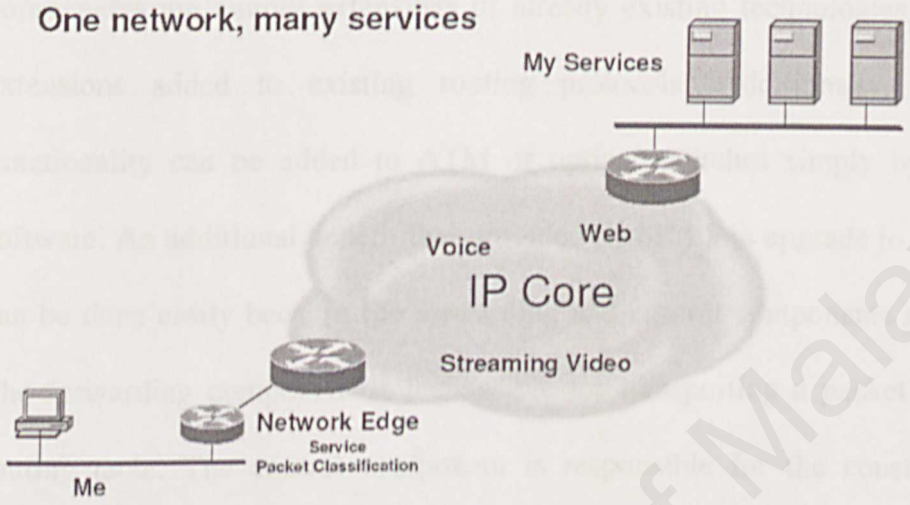


Figure 2.11: Basic MPLS architecture

MPLS components

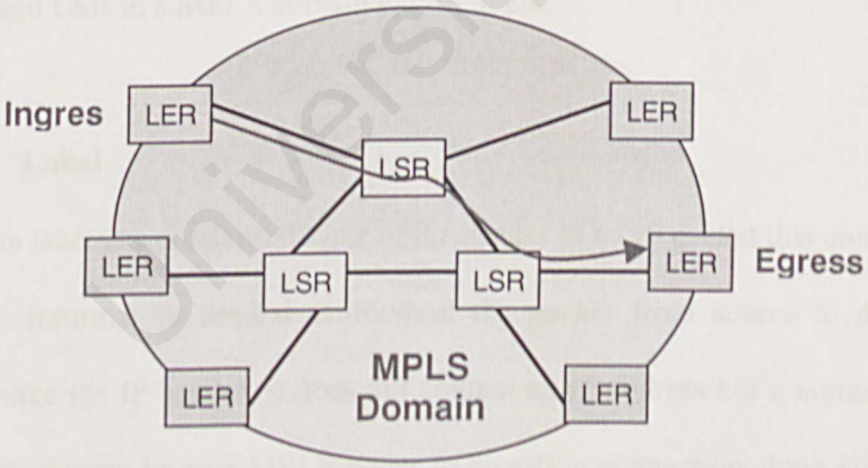


Figure 2.12: Basic MPLS domain environment

MPLS employs many new enhancements to IP routing in the forwarding of packets. Other components of the MPLS protocol enable it to function at a higher degree of performance and intelligence than current technologies. It also provides

a more efficient manner of forwarding packets from source to destination than the hop-by-hop basis used in traditional routing, as described earlier. Many of its components are simply extensions of already existing technologies such as the extensions added to existing routing protocols. Additionally, LSR/ LER functionality can be added to ATM or optical switches simply by upgrading software. An additional benefit that provided by MPLS is upgrade to the protocol can be done easily because the forwarding and control components are separate. The forwarding component is responsible for transporting a packet based on a routing table. The control component is responsible for the construction and maintenance of the routing table as well as working with the control components of other nodes to distribute routing information. Figure 2.12 the relation between LER and LSR in a MPLS domain environment.

i) Label

The label is a condensed view of the header of an IP packet that contains all of the information needed to forward the packet from source to destination. Unlike the IP header, it does not contain an IP address but a numerical value agreed upon by two MPLS nodes to signify a connection along an LSP. The label is a short, fixed-length, physically contiguous identifier that is used to identify a FEC. A packet assigned to a given FEC is usually based on its destination address either partially or completely. The label that is put on a particular packet represents the FEC to which that packet is assigned. Within

University of Malaya

some transport mediums there are existing labels that can be used by MPLS nodes when making forwarding decisions, such as ATM's virtual path identifier/virtual circuit identifier (VPI/VCI) field and frame relay's data link connection identifier (DLCI). Other technologies such as Ethernet and point-to-point links must use what is called a shim label, shown in Figure 2.13. The shim label is a 32-bit, locally significant identifier used to identify a FEC.

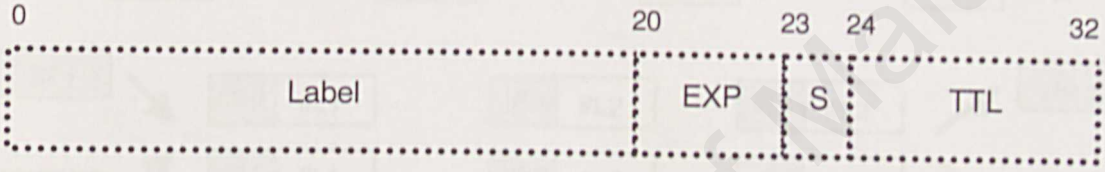


Figure 2.13: Format of Shim Label used in MPLS

- Label: 20 bits. A locally significant IS used to represent a particular FEC during the forwarding process.
- EXP: 3 bits. Previously called as class of service (CoS). It is now considered an experimental range. Currently, this field is being considered for QoS implementation.
- S: 1 bit. Used to signify if label stack is present. If the label is the only one present or at the bottom of the stack, the bit will be a value of zero.
- TTL: 8 bits. Field used to signify the number of MPLS nodes that a packet has traversed to reach its destination. The value is copied from the packets header and copied back to the header when it emerges from the ISP.

ii) Forwarding Equivalence Class (FEC)

By using FEC, any set of packets that are forwarded in the same way through a network. A FEC can include all packets that its destination address matches a particular IP network prefix or packets that belong to a particular application between a source and destination computer. Figure 2.14 shows concept of FEC.

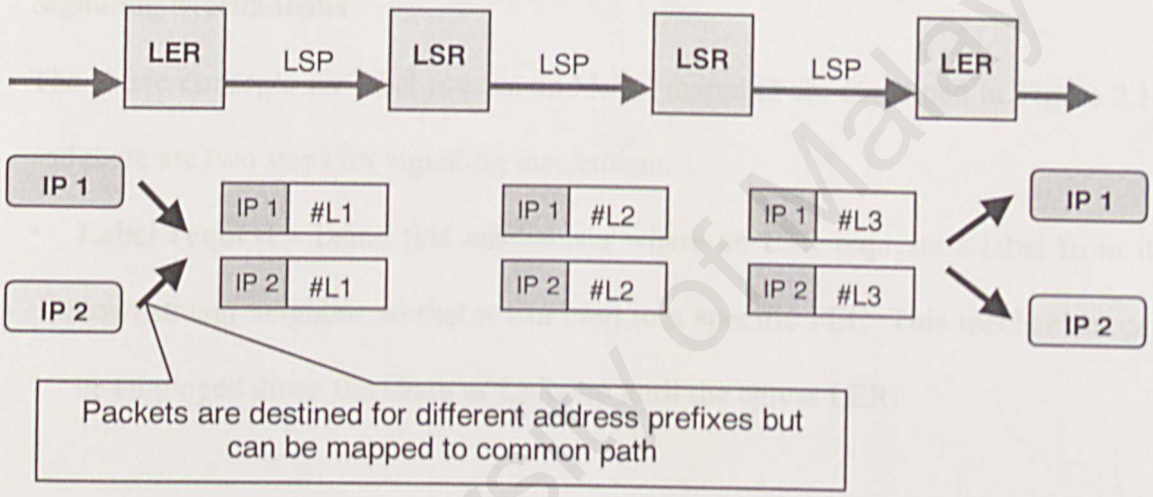


Figure 2.14: Forwarding Equivalence classes

iii) Label Switch Path (LSP)

In MPLS, data transmission occurs on label-switched paths (LSPs). LSPs are a sequence of labels at each and every node along the path from the source to the destination. The LSP is essentially the predetermined route that a set of packets bound to a FEC traverse through an MPLS network to reach their destination. Each LSP is unidirectional therefore return traffic must use a separate LSP.

iv) Label Stack

University of Malaya

By placing multiple labels onto a packet, MPLS can support a hierarchal routing design. The set of labels attached to a packet is called the label stack. As the packet traverses the network, only the top most labels are swapped. The labels are organized in a last-in first-out (LIFO) manner.

Signaling Mechanisms

The above concepts for label request and label mapping are explained in Figure 2.15 and there are two steps for signaling mechanism.

- **Label request** - Using this mechanism where an LSR requests a label from its downstream neighbor so that it can bind to a specific FEC. This mechanism can be employed down the chain of LSRs up until the egress LER.

- **Label mapping** - In response to a label request, a downstream LSR will send a label to the upstream initiator using the label mapping mechanism.

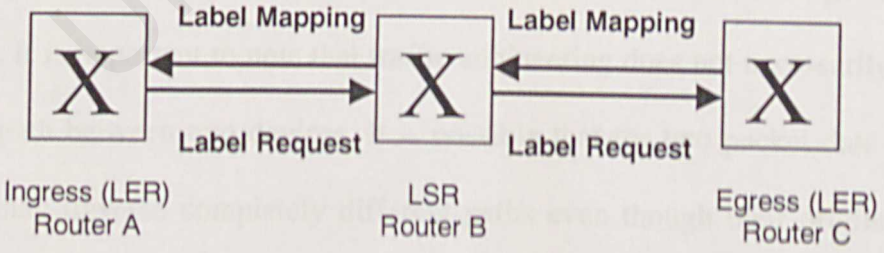


Figure 2.15: Signaling Mechanisms

Label-Switched Paths setup

A collection of MPLS enabled devices represents an MPLS domain. Within the domain, a path is set for a given packet to travel based on the FEC. The LSP is set prior to data transmission. MPLS provide two options to set up the LSP.

- **Hop-by-hop routing**

  Each LSR independently selects the next hop for a given FEC. This methodology is similar to that currently used in IP networks

- **Explicit Routing (ER)**

  Explicit routing is similar to source routing. ER-LSP follows route that source chooses ingress LSR.

Traffic Engineering

Traffic engineering is a process that enhances overall network utilization by attempting to create a uniform or differentiated distribution of traffic throughout the network. An important result of this process is the avoidance of congestion on any one path. It is important to note that traffic engineering does not necessarily select the shortest path between two devices. It is possible that for two packet data flows, the packets may traverse completely different paths even though their originating node and the final destination node are the same. The less-exposed or less-used network segments can be used and differentiated services can be provided.

MPLS Operation

University of Malaya

The following steps must be taken for a data packet to travel through an MPLS domain.

- Label creation and distribution
- Table creation at each router
- Label-switched path creation
- Label insertion/table lookup
- Packet forwarding

The source sends its data to the destination. In an MPLS domain, not all of the source traffic is necessarily transported through the same path. Depending on the traffic characteristics, different LSPs could be created for packets with different CoS requirements.

2.5 Network Simulator

2.5.1 Introduction to UMJaNetSim

The architecture of UMJaNetSim had shown in Figure 2.16. The basic concepts used by JaNetSim are summarized as follows:

- Discrete event model
- Central simulation engine with a centralized event manager.

University of Malaya

- Simulation consists of a finite number of interconnected components (simulation objects) and each of them has a set of parameters (component properties).
- Simulation execution involves components sending messages among each other. Scheduling an event for the target component sends a message.

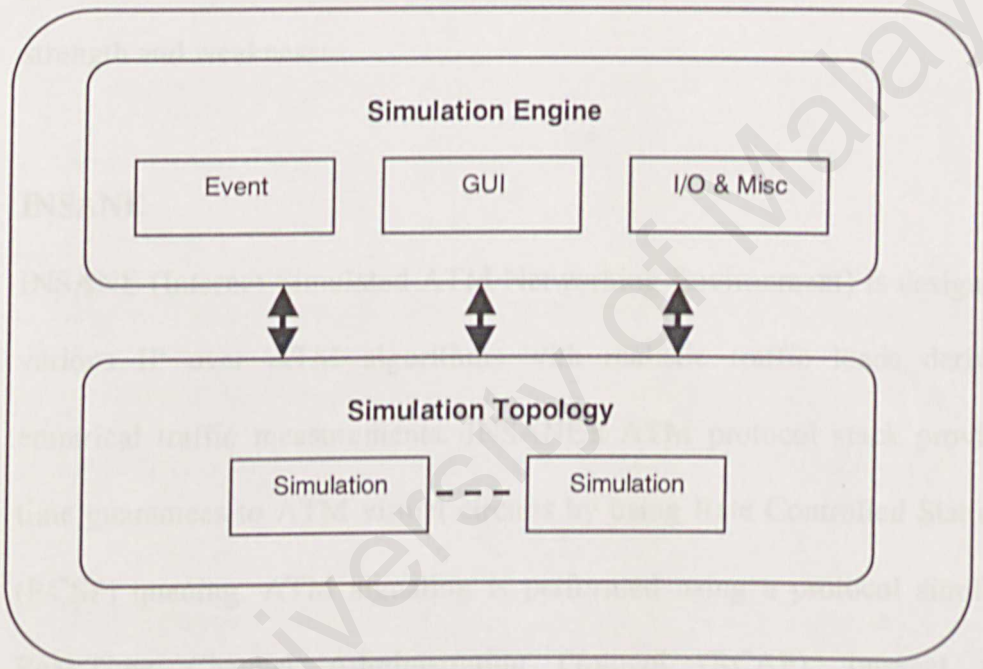


Figure 2.16: UMJaNetSim overall architecture

With the above architecture, the simulator can simulate virtually "anything" that can be modeled by a network of components that send messages to one another. These concepts are adopted from the NIST ATM/HFC Network Simulator.

The simulation engine is the sole event manager and is responsible for the managing of all the user interface elements. The engine also provides convenient means for file saving and data logging, among other tools. The programmer is

responsible for the development of simulation components that directly represent the intended system to be simulated.

2.5.2 Comparing for existing network simulator

The following are the current ATM network simulator evaluated to analysis their strength and weakness:

INSANE

INSANE (Internet Simulated ATM Networking Environment) is designed to test various IP over ATM algorithms with realistic traffic loads derived from empirical traffic measurements. INSANE's ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queuing. ATM signaling is performed using a protocol similar to the Real-Time Channel Administration Protocol (RCAP). Internet protocols supported include large subsets of IP, TCP, and UDP. In particular, the simulated TCP implementation performs connection management, slow start, flow and congestion control, retransmission, and fast retransmits. Various application simulators mimic the behavior of standard Internet applications to provide a realistic workload, including: telnet, ftp, WWW, real-time audio and real-time video.

University of Malaya

INSANE is designed to run large simulations whose results are processed off-line. It works quite well on distributed computing clusters (although simulations are all sequential processes, a large number of them can easily be run in parallel). Although there is no graphical user interface, a (optional) Tk-based graphical simulation monitor provides an easy way to check the progress of multiple running simulation processes. The bulk of INSANE is written in C++. Customization and simulation configuration is performed with Tcl scripts.

Advantages

The Tk-based graphical simulation monitor enable user to check the progress of multiple running simulation process. Besides that, it is able to support the simulation on a large network, which the result is processed off-line.

Disadvantages

The simulator can only works on a few hardware and platforms only and this restricted the portability of the simulator. Furthermore, there are a few software requirements to run the simulator and this will be troublesome for the user to use the software.

NIST ATM/ HFC

This simulator was developed at the National Institute of Standards and Technology (NIST) and it is a tool to analyze the behavior of ATM and HFC

University of Malaya

networks without the expense of building a real network. Therefore, this simulator can conceivably be used to plan be used to plan ATM networks as well as analyze ATM and HFC protocols. It allows the user to interactively model the environment with a graphical user interface.

By using the NIST ATM/HFC simulator, the user can create different network topologies; adjust the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data simulation execution.

Advantages

The user can create different topologies and able to adjust the parameters during the simulation of the network. The user can save and load various simulation configuration. The simulator provides a graphical user interface and enable user to drag and drop the entities in the network.

Disadvantages

Users of the simulator might face problems setting up the network topology because they need to input a large number of parameters. The customization of the simulator's component requires user or programmers to have strong foundation in C. Besides that, it is using procedural approach whereby the

University of Malaya

components have overlapped functions between the components. The simulator only can run on limited platform that is UNIX and LINUX platform.

YATS

YATS (Yet Another Tiny Simulator) is a small cell-level simulation tool for ATM. Its kernel comprises the event scheduler, a symbol manager and a scanner/parser front end. An input file describes the arbitrary model network configuration, the simulation actions and the way to analyze the results. The input language is a simple script language, which allows for a flexible problem description (loops, macros and basic mathematical capabilities are provided). The discrete time event scheduler applies a static calendar queue and unusual event memory management, which results in good simulation speed.

The system is written in C++. All network nodes are objects that communicate over standardized messages. Graphical object classes are able to display the time dependent behavior of variables and distributions inside of other model objects (without adding complexity to these network objects).

Advantages

The simulation of the network has reasonable speed and simple models virtually can run in real time. The simulator has high flexibility of integrated model description, simulation control, and result analysis. The whole

University of Malaya

simulation experiment can be instrumented via environment variables that in turn allows - together with a shell script - to easily perform complete experiment series over night. Although it is very simple, the online displays are useful to understand what's happening in the model network. This especially holds for ABR, TCP and all this protocol stuff.

Disadvantages

The pure slotted operation causes some restriction when simulating different line speeds in the same model. It's only possible to choose speeds for which the cell transfer time is an integer multiple of a basic time used for the whole model. The multiplexer objects classes MuxAF/ MuxDF emulate lower line speeds. The ABR multiplexer does not yet support lower speeds. The discrete-time nature excludes some useful source models like the Poisson types. While the language based model description yields a high flexibility, the input may become a bit irritating in case of larger networks.

OMNET++

OMNeT++ (Objective Modular Network Testbed in C++) is a discrete event simulation tool. It is primarily designed to simulate computer networks, multi-processors and other distributed systems, but it may be useful for modeling other systems tool. OMNeT++ has been developed on Linux, but it works just as well on most Unix systems and on Windows platforms (NT recommended). It provides

University of Malaya

a simulation library with statistical classes and an environment that supports interactive simulation including the visualization of collected data. The gnu plot-based GUI tool is used for analyzing and plotting simulation results.

Advantages

OMNeT++ has a solid and flexible simulation kernel and it provide powerful GUI environment for simulation execution. Users can build hierarchical and reusable models easily. The interface is human readable and its source code is provided.

Disadvantages

User must use command line to simulate the network and posses knowledge in C or C++ programming languages to use OMNeT++.

NetSim++

In a nutshell, NetSim++ is a software package designed to provide a comprehensive work environment for the network modeler. It can be used in areas of communications networks such as performance measurement for existing or future networks under a wide range of conditions. Besides that, it can perform analysis and simulation of queuing systems. NetSim++ is designed specifically for the development and analysis of communications networks. Models can be hierarchically structured, allowing their re-use in different simulations.

University of Malaya

Specifications are entered graphically with specialized editors. The editors provide an efficient medium for design capture via a consistent set of modern user-interface elements. NetSim++ follows for the hierarchy and communication model a subset of SDL-92 semantics. As with SDL, the active parts are processes; a hybrid approach is used to embed C++ language code with a graphically specified Extended Finite State Machine (EFSM).

Advantages

NetSim++ provides an efficient event-driven Simulation Kernel, a Simulation API and a Base Models Library of components. It takes the design specification and automatically generates an executable simulation. A set of analysis tools is provided to interpret and visualize a large volume of simulation results.

Disadvantages

The current implementation of NetSim++ is available only for UNIX/X Window System platforms.

University of Malaya

Summary of comparing existing network simulator

Based on the evaluation of the network simulator, the Table 2.2 below compare the network simulator on a few feature such as discrete-event simulator, object-oriented, GUI, multithreaded, web enabled, platform independent.

Simulator	Discrete Event Simulation	Object-oriented	GUI	Multithread	Web-enable	Platform independent
INSANE	√	√	√	√	X	X
NIST ATM/ HFC	√	X	√	X	X	X
YATS	√	√	X	√	X	X
OMNET++	√	√	√	X	X	X
NetSim++	√	X	√	√	X	X

Table 2.2: Comparison among various simulators

2.6 Chapter Summary

This chapter has covered the primary research background of this project and relevant knowledge needed to develop the network simulator. A more detailed explanation of the simulator will be presented in the following chapter.

University of Malaya

CHAPTER 3: TRAFFIC CONDITIONING

There are few methods to be used in order for implementing traffic conditioning into UMJaNetSim that are:

i) Analysis

This method will use a lot of calculations that involve many complex mathematic formulas. Complex mathematic formulas have to apply by using parameters that set by user and predict the output for the flow. This is not a sufficient and effective ways since this method is complex and user cannot have a clear picture for the path and properties for the entire component. The result of analysis is not accurate since the answer will be rounded.

ii) Real time application

This method will involve many real components which is becoming a problem to locate all the components and all the properties for the components is had to be change once the components had been set it. Just consider that you are doing a simulation which includes 5 routers and 10 switch and 20 links, how can you allocate these entire components and how to change its setting for different testing results?

iii) Simulation

This method is better comparing to the previous methods and it had been chosen to implement network simulator. This is a method which is involving all the virtually set components such as routers, switches and link into our simulation.

University of Malaya

This is an effective and efficient way since the component's setting and arrangement can easily changed in order to achieve best performance for the simulation results.

3.1 Traffic Conditioner

A Traffic Conditioner mechanism usually applied at the network boundary to ensure the traffic flow following the Traffic Conditioning Agreements (TCA) rules. TCA is required for implementing DS for network traffic management. The TCA (Cisco 2003):

- Describes traffic characteristics or also known as traffic profiles for the traffic flow being sent and received by the connection.
- Defines policies governing the amount and kind of traffic that can be sent and received by the connection.
- Lists down the Differentiated Services (DS) rules to follow in order to remain to the TCA, such as re-marking the DSCP or dropping packets.

The Traffic Conditioner measures the input traffic and assures that the packet behavior follows the predefined profiles. The DiffServ primarily conditions the traffic at the edge router only. In order to implement Traffic Conditioner into UMJaNetSim, Token Bucket and Leaky Bucket algorithm had been chosen.

University of Malaya

First of all, a classifier reads the DSCP value from DS fields and selects the packets and routes packets to a Traffic Conditioner. Figure 3.1 shows how the Traffic Conditioner handles the traffic flow and ensures the traffic following TCA rules. (Juha 2002)

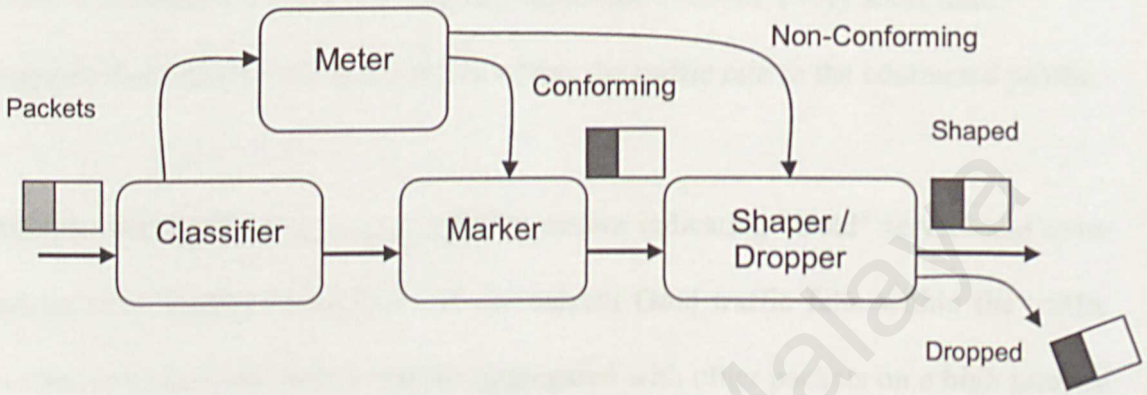


Figure 3.1: Traffic Conditioner ensures the TCA is remain

Like a police officer, a Traffic Conditioner may be needed to monitor and control the traffic flow into the downstream DS domain by:

a) **Meter** compares the stream behavior of packets selected by the Classifier with the traffic profile specified by the Traffic Conditioning Agreement (part of SLA).

b) **Re/Marker** sets a specific value in the DS field of a packet, and adds the marked packet to the specific DS Behavior Aggregate.

c) **Shaper** delays part or all of packets to adapt the packet stream to the contracted traffic profile. A leaky bucket model or a pure shaper can be applied. The leaky bucket model shapes the traffic according to the token rate and depth of the bucket. Although the sustainable rate of traffic is limited by the token rate, the packet burst can be allowed according to the depth of the bucket even when the instantaneous

University of Malaya

traffic rate exceeds the token rate. In contrast, the pure shaper has a Leak Rate parameter that actually limits the peak rate during packet egress processing. No packet is allowed to violate this leak rate limitation even for a very short time.

d) **Dropper** discards packets and forcibly adapts the traffic rate to the contracted profile.

For example, the classifier may read a packet stream indicating "Gold" service and route the packets to a Traffic Conditioner. If the current Gold traffic falls within the traffic profile characteristics, the packet may be aggregated with other packets on a high priority route and sent immediately to the next hop. However, if the packet stream is out of profile, the packet DSCP may be changed to map the packet to a "Silver" service level or the packet may be re-queued or dropped altogether according to the TCA rules. Alternatively, "lower class" packets may by remarked or dropped according to the policy. A Traffic Conditioner may not include all these functions. Only the functions needed to adhere to the TCA policies would be implemented.

University of Malaya

3.2 Token Bucket

3.2.1 Traffic Policing / Metering

Generally, token bucket represents the policing and shaping function of Traffic Conditioning in a Diffserv environment. A token bucket also used to manage a device that regulates the data in a flow. In Figure3.2, the regulator might be a traffic policer. Traffic policing controls the maximum rate of traffic sent or received on an interface. Based on the results of the token bucket measurement, an action can be configured to mark packets and separate packets into multiple classes or levels of service. (Ion 2003)

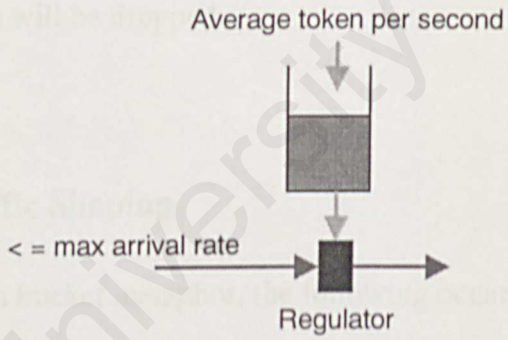


Figure 3.2: Regulator in Token Bucket act as a traffic policer

Traffic Policer provides two main benefits:

- Bandwidth management through rate limiting
- Allow user to control the maximum rate of traffic sent or received on an interface. Traffic policing is often configured on interfaces at the edge of a network to limit traffic into or out of the network. Traffic that falls within the

University of Malaya

rate parameters is sent, whereas traffic that exceeds the parameters is dropped or sent with a different priority.

- Packet marking through IP precedence, QoS group, or DSCP value setting
- Packet marking allows user to partition the network into multiple priority levels or classes of service (CoS). Use traffic policing to set the IP precedence or differentiated services code point (DSCP) values for packets entering the network. Networking devices within your network can then use the adjusted IP Precedence values to determine how the traffic should be treated. For example, the VIP-Distributed Weighted Random Early Detection feature uses the IP precedence values to determine the probability that a packet will be dropped.

3.2.2 Traffic Shaping

In the token bucket metaphor, the following occurs:

- Tokens are put into the bucket at a certain rate. The bucket has a specified capacity.
- If the bucket full of token, newly arriving tokens are discarded.
- Each token has permission for the source to send a certain number of bits into the network.
- In order to send a packet the regulator must remove enough tokens from the bucket which is equal to the packet size.

University of Malaya

- If not enough tokens are in the bucket to send a packet, the packet either waits until the bucket has enough tokens or the packet is discarded or marked down.
- If the bucket is already full of tokens, incoming tokens overflow and are not available to future packets. Thus, at any time, the largest burst that a source can send into the network is roughly proportional to the size of the bucket.

The token bucket mechanism used for traffic shaping has both a token bucket and a data buffer or queue. If the token bucket mechanism for traffic shaping did not have a data buffer, it would be a traffic policer.

3.2.3 Token Bucket Model

Figure 3.3 shows above the new tokens are adding to the bucket at rate of r tokens/sec, the maximum token can be accumulated is b bytes. If the bucket is full, the incoming tokens will be thrown away. (Chimento 2003) The Token Bucket profile contains three major parameters:

1) **Average Rate,**
2) **Peak Rate and**
3) **Burst Size**

University of Malaya

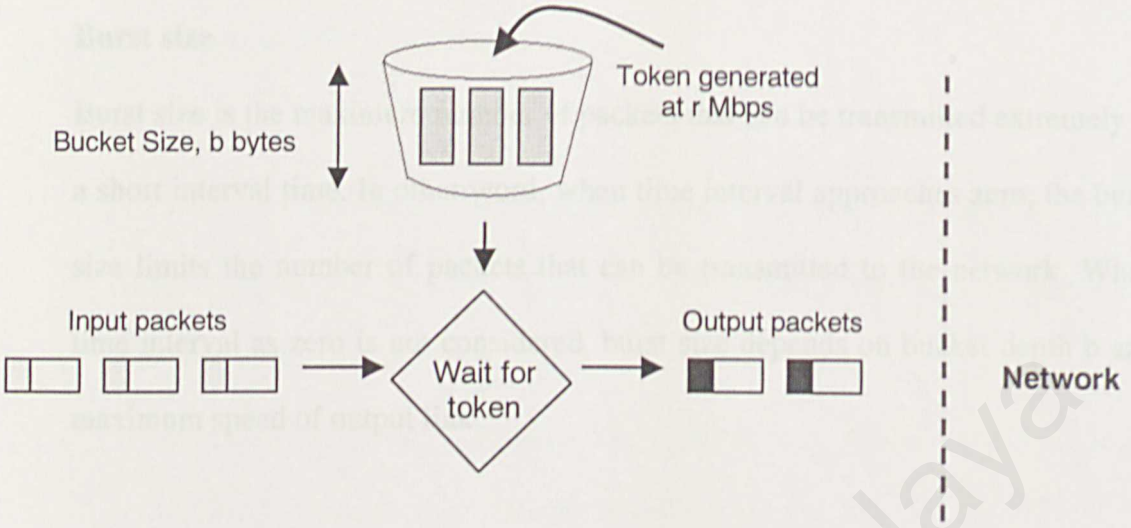


Figure 3.3: Concept of using Token Bucket as a Traffic Conditioner

Average rate

The network may wish to limit the long-term average rate at which packets in a flow can be sent into the network. The important point here is the interval of time over which the average rate will be policed. For example, a flow which has 10000 packets/s is more flexible than a flow which has 10 packets/ms. Even though both have the same average rate, their behaviors are different. The flow which has 10000 packets/s can send 100 packets in a one ms long-term, but the flow which has 10 packets/ms cannot send 100 packets in a one ms long-term. This is one of the important features we should consider.

Peak rate

Defines the maximum rate at which packets can be sent in short interval time. In above example; even though the flow's long term average rate is 10000 packets/s peak rate can be limited to 200 packets/ms.

University of Malaya

Burst size

Burst size is the maximum number of packets that can be transmitted extremely in a short interval time. In other word, when time interval approaches zero, the burst size limits the number of packets that can be transmitted to the network. When time interval as zero is not considered, burst size depends on bucket depth b and maximum speed of output link.

The time which bucket gets empty is (B + rS = pS). One of the potential problem with token bucket is it allows burst again. If another burst comes during this time interval, it cannot be handled or in the case of bursts come very often the output speed will be the maximum for the period of burst that can cause congestion. This problem can be reduced by carefully choosing parameters. Essentially, the network has to simulate the algorithm and make sure that no more packets or bytes are being sent than are permitted. Depends on the agreements the excess packets can be dropped, remarked etc. (J.Clasmann, 2003)

University of Malaya

3.2.4 Example of Token Bucket:

Case:

- Tokens generated with rate r where one token equal to one packet.
- Packet must wait for a token before transmission where it is no losses and allows limited bursts.
- When packets are not generated, tokens accumulate where n tokens is ready for burst of n packets and if bucket filled, tokens are lost
- Mean departure rate is r

Therefore

Burst duration - *S* sec

Size of the bucket - *B* bytes

Maximal departure rate - *p* bytes/s

Token arrival rate - r bytes /s

The time which buckets get empty

$$B + rS = pS$$

Where

$$S = B / (r-p)$$

Example:

B = 250 Kb, p = 25 Mb/s, r = 2 Mb/s

$$B+rS = pS$$
$$S = B/(r-P)$$
$$= 250\,000 / (2\,000\,000 - 25\,000\,000)$$
$$= -0.01087s$$
$$S = 11 ms$$

University of Malaya

3.2.5 Token Bucket in Diffserv

The operation of the Token Bucket in diffserv can be defined as follow:

- Arriving packets of L bytes are conforming (immediately processed) if there are at least L tokens in the bucket (one token= one byte)
- If the current number of accumulated tokens b' is less than the arriving number of packets L, L-b' packets are nonconforming.
- Packets are allowed to the average rate in bursts up to the burst size as they do not exceed the peak rate at which point the bucket is drained.
- If there are no packets to be transmitted, tokens can be accumulated up to size of b. The rest of tokens will be thrown away.

Conforming and Non-Conforming functions are depends on SLA agreement. If the packets are non-conforming the following actions can be taken.

- The packet may be thrown away.
- The packet may be re-marked in a particular way.
- The packet can be buffered (by inserting a buffer between the inflow and the decision point) and not released until sufficient numbers of tokens arrive in the bucket.
- These functions are applied to different classes in different ways.

University of Malaya

3.3 Leaky Bucket

3.3.1 Traffic Policing

There are two types of leaky bucket used for traffic policing (Nikos, 2002):

i. Single Leaky Bucket

ii. Double Leaky Bucket

Single Leaky Bucket

In single leaky bucket, one leaky bucket is used to police traffic single parameters. Cells that are conforming to this leaky bucket 1 are admitted to network. Figure 3.4 shows the Single Leaky Bucket can be used as a traffic policer.

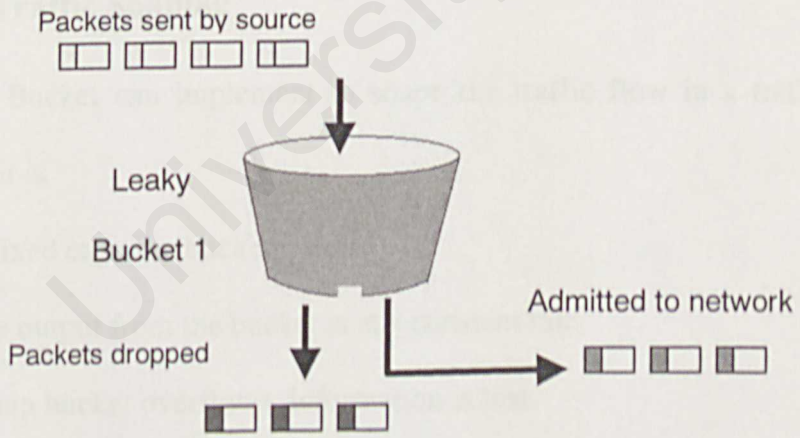


Figure 3.4: Single Leaky Bucket as a traffic policer

Double Leaky Bucket

Two leaky buckets are used in double leaky bucket. Each leaky bucket controls different parameters in the simulator. Cells that are conforming to both leaky

buckets are admitted to the network. Figure 3.5 shows the Double Leaky Bucket can be used as a traffic policer.

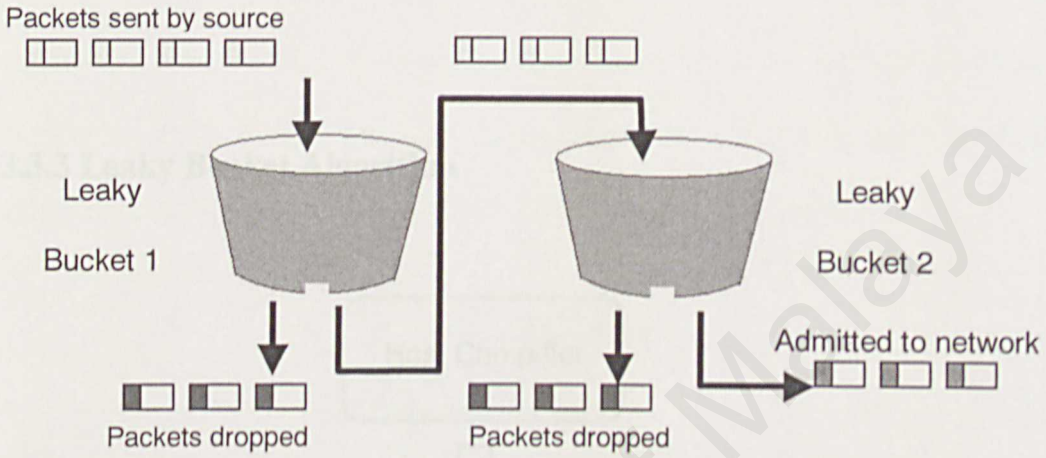


Figure 3.5: Double Leaky Bucket as a traffic policer

3.3.2 Traffic Shaping

Leaky Bucket can implement to shape the traffic flow in a traffic conditioner where it is

- A fixed capacity bucket
- The output from the bucket is at a constant rate
- When bucket overflows, information is lost.

A leaky bucket is actually a bucket with a fixed capacity that has a hole in the bottom by leaking out data with constant rate. It can be thought of a single server queue with finite buffer and constant service time which is equivalent to a single server queuing system with constant service time. Moreover one packet (for

fixed-size packets) or a number of bytes (for variable-size packets) are allowed into the queue per clock cycle. Congestion control is accomplished by discarding packets arriving from the host when the queue is full.

3.3.3 Leaky Bucket Algorithm

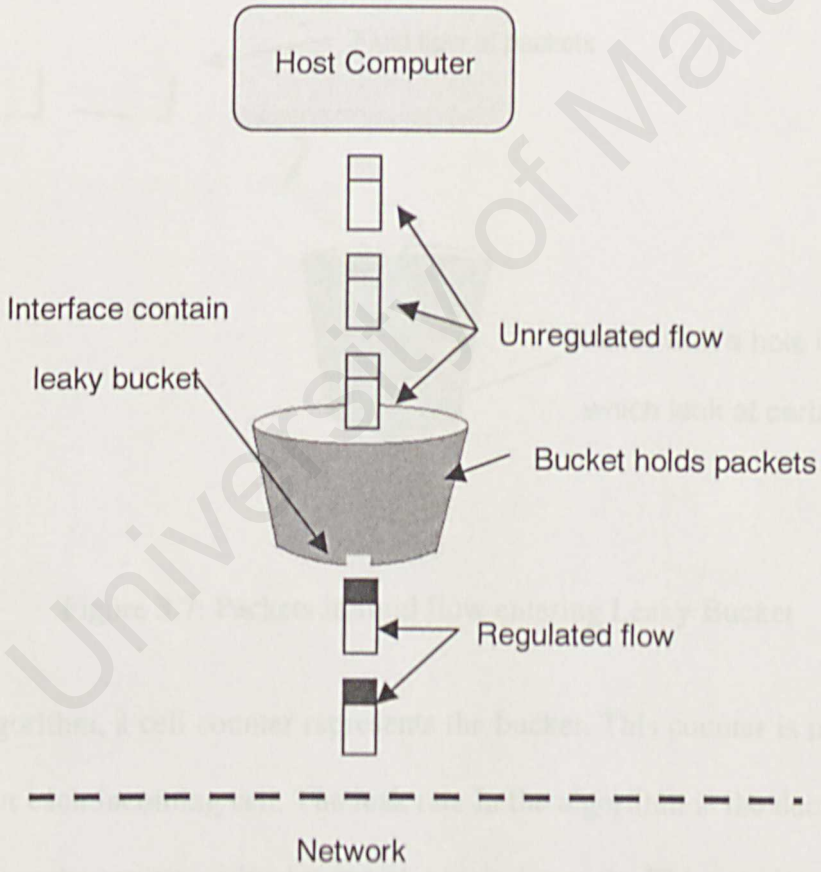


Figure 3.6: Concept of using Leaky Bucket as a Traffic Conditioner

Figure 3.6 shows the concept of using Leaky Bucket as a Traffic Conditioner. The leaky bucket algorithm is a key to define the meaning of conformance. The leaky

University of Malaya

bucket similarity refers to a bucket with a hole in the bottom that leak at a certain rate corresponding to a traffic cell rate parameter. The depth of the bucket corresponds to a tolerance parameter. Figure 3.7 shows that each cell arrival creates a cup of fluid flow poured into one or more buckets for use in conformance checking. The Cell Loss Priority (CLP) bit in the cell header determines which bucket the cell arrival fluid pours into. (Rad 1999)

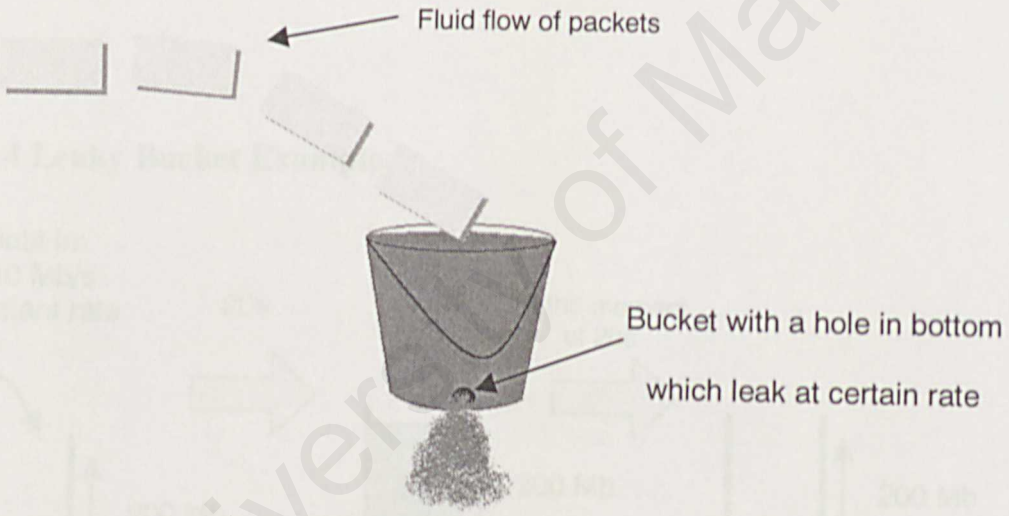


Figure 3.7: Packets in fluid flow entering Leaky Bucket

In the algorithm, a cell counter represents the bucket. This counter is incremented by one for each incoming cell. The leak rate in the algorithm is the decrement rate that reduces the counter value by one at certain intervals. This rate is given by the cell rate under consideration and is governed by the minimum distance between two consecutive cells. The bucket volume is corresponding to the cell counter range which is represented by the permissible time tolerance for the incoming cells. This value is determined through the traffic contract or is set by the network

provider and is called CDVT (cell delay variation tolerance). If the counter exceeds a certain value, the cells are assumed not to conform to the contract. To counteract this, non-conforming cells can now either be tagged or dropped. The algorithm is called dual leaky bucket if several parameters are monitored at once or single leaky bucket if only one parameter is monitored. In the Leaky Bucket analogy the cells do not actually flow through the bucket but it only the check for conformance to the contract does.

3.3.4 Leaky Bucket Example

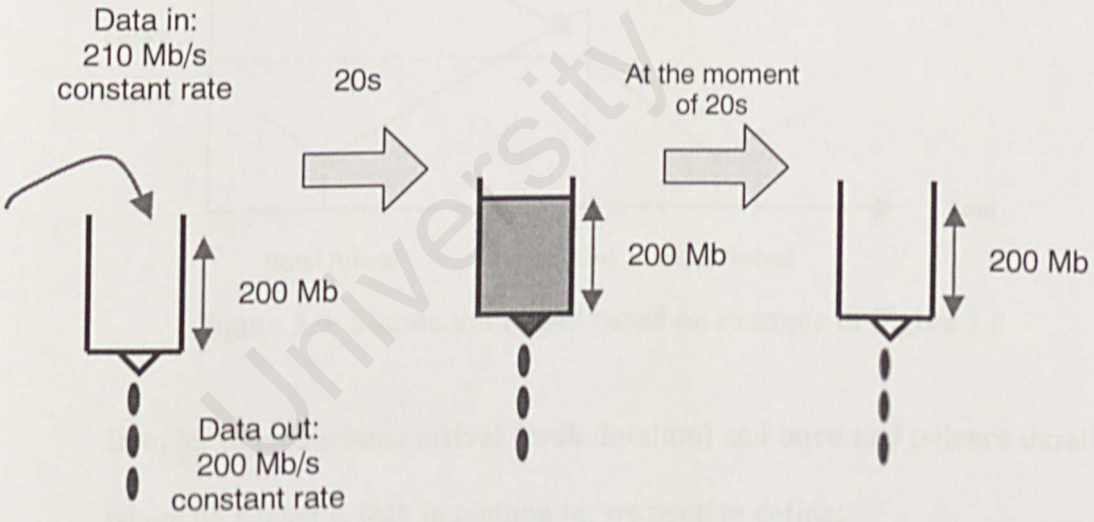


Figure 3.8: Leaky Bucket Example

Figure 3.8 shows an example of leaky bucket. Assume that a leaky bucket with bucket size 200 Mb that allow data flow in with constant rate, 210 Mb/s and data will leak out with constant rate as well 200 Mb/s. At the moment of 1 second, the whole bucket will be full of data and the bucket will start to leak out at constant

University of Malaya

rate, 200Mb/s. Therefore, there are 10 Mb data accumulated in the bucket. This process is continuous until 20 second after starting, the accumulated data in bucket will be 200 Mb (10 Mb accumulated for every 1 second). At the moment of 20 second, the accumulated data in the bucket will leak out at once even no incoming data from the flow. The simulation result is shown in Figure 3.9.

Simulation result:

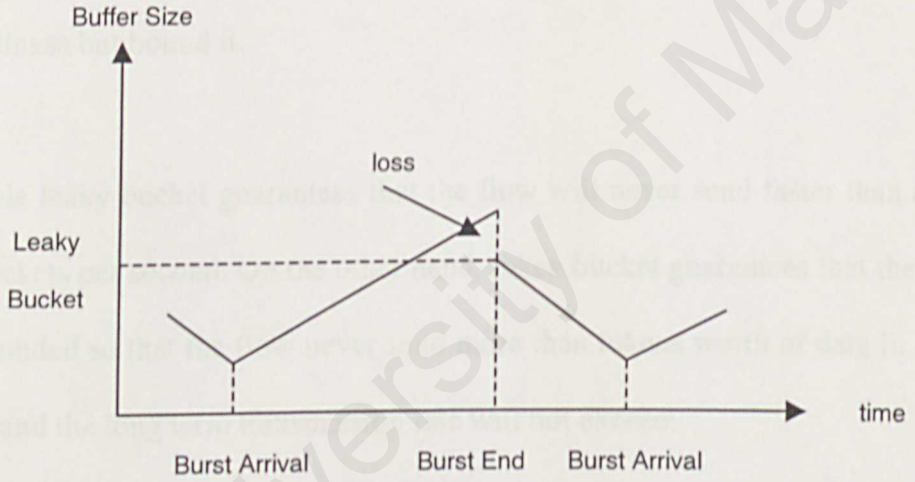


Figure 3.9: Simulation Result based on example of Figure 3.8

In order to create burst arrival (peak duration) and burst end (silence duration) where no packet is leak in coming in, we need to define:

a) t_{on} – the average duration of the on-state (peak duration)

b) t_{off} - the average duration of the off-state (silence duration)

University of Malaya

3.4 Differences between Leaky Bucket and Token Bucket

- Token bucket throws away token but never buckets.

- Token bucket has no discard or priority policy.

- Simple leaky bucket forces bursty traffic to smooth out while token bucket permits burstiness but bound it.

- Simple leaky bucket guarantees that the flow will never send faster than total worth of packets per second. On the other hand, token bucket guarantees that the burstiness is bounded so that the flow never send more than tokens worth of data in an interval time and the long term transmission rate will not exceed.

3.5 Chapter Summary

This chapter had explained the approach for Traffic Conditioner and how it works. Besides that, both Leaky Bucket and Token Bucket had been explained clearly. Finally, the comparison between Leaky Bucket and Token Bucket in stated.

University of Malaya

CHAPTER 4: ANALYSIS

4.1 Simulator Platform

With the rapid development on Windows system, the processing speed of a PC runs on Windows is comparable with a UNIX system and the price of a PC that runs on Windows platform is lower compare to a UNIX workstation, therefore the Windows platform becomes dominant at the recent days. Although the Windows platform becomes more common, the UNIX platform cannot be neglected.

Most of the currently available simulator is running on UNIX platforms with only a few that run on Windows platform. A network simulator that is cross platform is more important as it is supported by most of the platform such as UNIX or Windows.

4.2 Programming Language

Several programming approaches can be taken as approach to the development of the ATM network simulator. Therefore it is a need to consider many advantages and disadvantages of several programming approaches. Here, procedural programming and object oriented programming are both discussed.

University of Malaya

4.2.1 Procedural Programming

The procedural approach makes use of procedural languages in which program codes were placed into blocks that are referred to as procedures or functions. With the use of procedural languages, tasks were broken down into separate blocks in which separate blocks would perform separate tasks.

Procedural Programming specifies an exact sequence or order of operation where procedural program is written as a list of instructions telling the computer step-by-step what to do. Procedural programming is fine for small projects. It is the most natural way to tell a computer what to do and the computer processor's own language, machine code is procedural so the translation of the procedural high-level language into machine code is straightforward and efficient. The procedural programming has even built-in way of splitting big lists of instructions into smaller lists.

4.2.2 Object-oriented Programming

Object-oriented Programming (OOP) is different from procedural programming language in several ways. Everything in OOP is grouped as object. Every object communicates with each other by sending message. Objects are central idea behind OOP. The basic idea behind an object is that of simulation. In object-oriented methodology, a program should be written to simulate the states and activities of real world objects. OOP has its key component technologies

University of Malaya

inheritance and polymorphism. Inheritance is a form of software reusability in which new classes are created from existing classes by absorbing their attributes and behaviors and embellishing these with capabilities the new classes require. Polymorphism is a character of assigning different meanings to a particular symbol or object depending upon the context in which it is used. This allows objects to act differently within different situations, it enable the flexibility of a program design.

The following describe some advantages of object-oriented programming language:

- Simplicity
- Modularity
- Modifiability
- Extensibility
- Maintainability
- Reusability

Java Programming

Java is just a small, simple, safe, object-oriented, interpreted or dynamically optimized, byte-coded, architecture-neutral, garbage-collected, multithreaded programming language with a strongly typed exception-handling mechanism for writing distributed, dynamically extensible programs. Java is developed by Sun

University of Malaya

Microsystems in California. It is a powerful programming language built to be secure, cross-platform and international.

As in a modern software development, Java is object-oriented language which was designed for use in Internet environment. Java can be used to create complete applications that may run on a single computer or to be distributed among servers and clients in a network. It can also be used to build applets for use as part of a web page.

The major advantages of Java are simple, object-oriented, network-savvy, robust, secure, architecture neural, portable, interpreted, high performance, multithread, dynamic

C++ Programming

Bjarne Stroustrup developed C++, an extension of C in the early 1908s at Bell Laborites. C++ provides a number of features that "spruce up" the C language but more importantly, it provides capabilities for object-oriented programming. C++ is a hybrid language. The C++ language facilities structured and disciplined approach to computer program design. C++ programs consist of pieces called classes and function Programmer can program each pieces that programmer need to form a C++ program. But most C++ programmers take advantage of the rich collections of existing classes and functions in the C++ standard library. C++

University of Malaya

programs typically go through six phases to be executed as shown in Figure 4.1. These are: edit, preprocess, compile, link and execute.

Java Programming Tools

There are many types of Java programming tools available in market. One can develop Java programming using pure Java SDK without the supporting of integrated development environment (IDE) or just selects tools like Microsoft J++, JCreator, Borland JBuilder or Symantec Visual Café.

University of Malaya

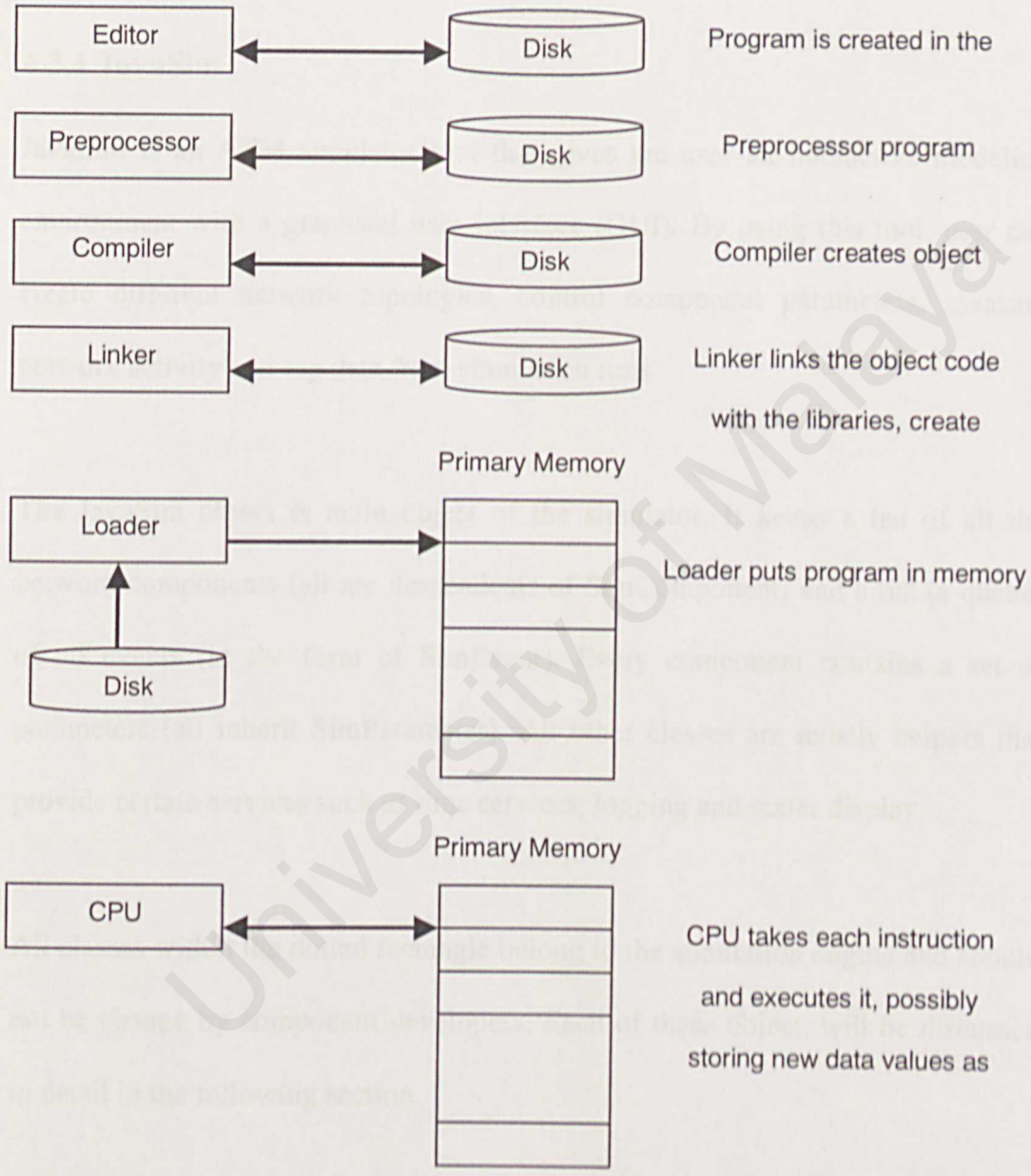


Figure 4.1: Typical C++ Environment

University of Malaya

4.3 Analysis of UMJaNetSim

4.3.1 JavaSim

JavaSim is an ATM simulator tool that gives the user an interactive modeling environment with a graphical user interface (GUI). By using this tool, user can create different network topologies, control component parameters, measure network activity and log data from simulation runs.

The JavaSim object is main object of the simulator. It keeps a list of all the network components (all are descendents of SimComponent) and a list (a queue) of all events (in the form of SimEvent). Every component contains a set of parameters (all inherit SimParameter). All other classes are mostly helpers that provide certain services such as time services, logging and meter display.

All classes within the dotted rectangle belong to the simulation engine and should not be change by component developers. Each of these object, will be discussed in detail in the following section.

University of Malaya

Figure 4.2 showing the Hierarchy of all the significant objects in the UMJaNetSim

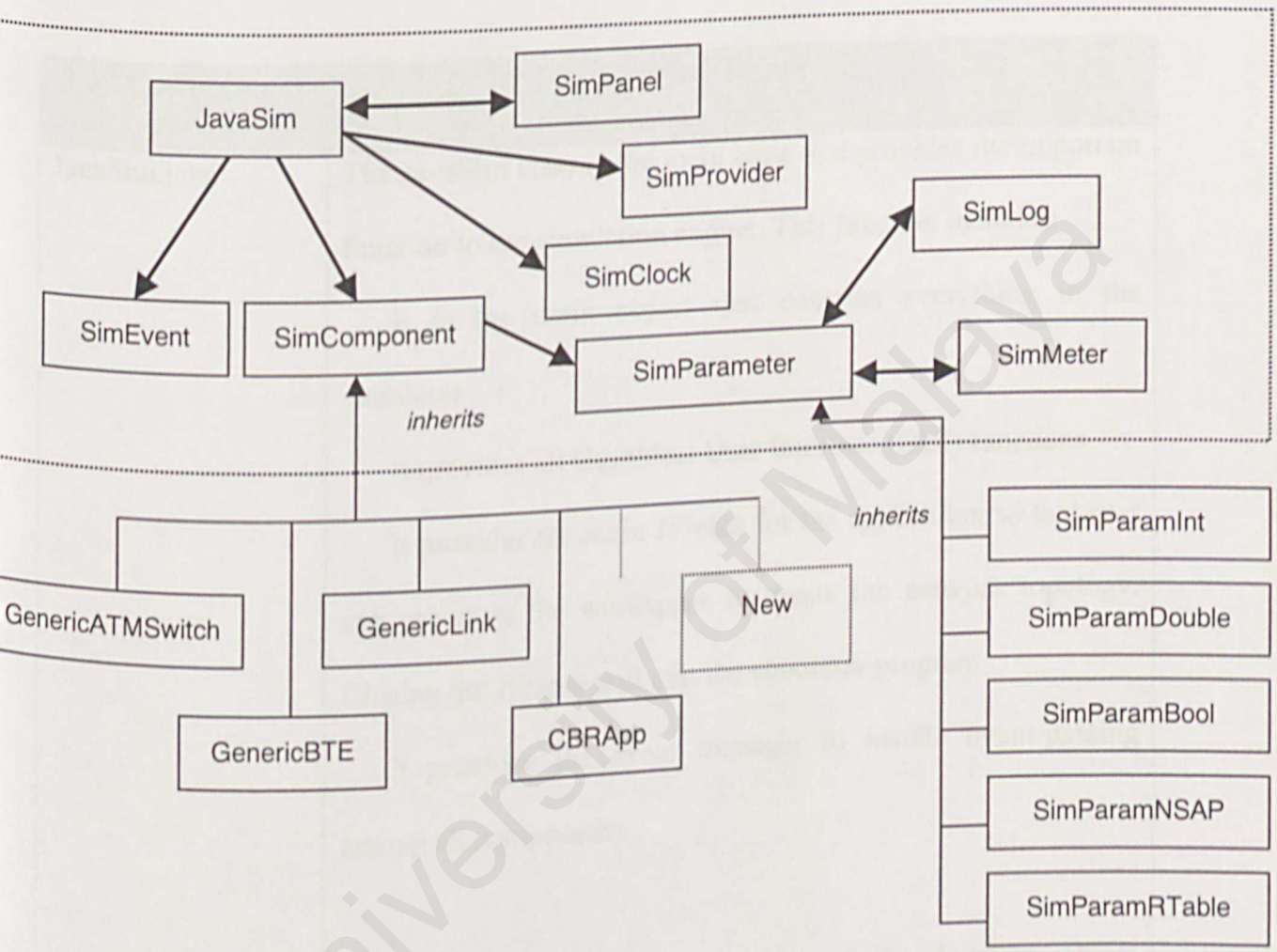


Figure 4.2: Hierarchy of all the significant objects in the UMJaNetSim

University of Malaya

4.3.2 Simulation Engine

Table 4.1 describing major functions for java files including in UMJaNetSim.

Object	Major Functions
JanaSim.java	The JavaSim class is the main class that provides the important function to the simulation engine. This function includes: It is the main object that contains everything in the simulator It provides all Graphical User Interface (GUI) functions It provides the main JFrame for the application so that user can use it as the workspace to create the network topology. Closing the JFrame will exit the simulator program It provides the event manager to handle event-passing among all components There will be only one interface of the JavaSim object throughout the simulation. Anyone with a reference to this instance can make use of the following services : *long now();* *//this function return current simulation time in tick* *java.util.List getSimComponents();* *//it returns a list of all existing SimComponent* *boolean isCompNameDuplicate(String name);*

University of Malaya

	/*this function returns true if the supplied parameter name is already used by another*/
	/*SimComponent it prevents two components with the same name happens*/
	void notifyPropertiesChange(SimComponent comp);
	/*SimComponent must call this whenever there are structural changes to the parameters, for example: add or remove parameters*/
	void enqueue(SimEvent e);
	/*Every communication (message exchange) between any components must involve creation of a SimEvent and call to the enqueue( ) method */
	void dequeue(SimEvent e);
	/*it removes a SimEvent from the queue
	/*the SimEvent parameter must be one that has been enqueue into the queue before*/
	
	Objects that have a reference to the main JavaSim object can only call these functions.

Cell.java	The cell class is a data resource class used by components like CBRApp, VBRApp, GenericBTE and GenericATMSwitch throughout the simulator. As a result, it contains attributes needed by the operation of all executor classes.

University of Malaya

	The main parameters in cell class are:
	int vpi=0; //virtual path identifier
	int vci=0; //virtual channel identifier
	int pti=0; //payload type identifier
	int CLP=0; //cell loss priority
	
	At the initial stage, vpi, vci, pti and CLP values are set to zero. The payload type identifier is used by the ATM end system to determine how to handle incoming cells. The pti value can be 0,1,2 or 3
	// 0 – last data cell
	// 1 – not last data cell
	// 2 – forward RM cell
	// 3 – backward RM cell

SimClock.java	The SimClock class provides a set of time translation functions for normal translation between tick and actual time (in microseconds, milliseconds and seconds). It is an importance class to synchronize the time throughout the simulation process. The functions provided by SimClock class is as follows: *static double Tick2Sec(long tick);* *//converts ticks to seconds* *static double Tickc2MSec(long tick);*

University of Malaya

	//converts ticks to milliseconds; *static double Tickc2USec(long tick);* *//converts ticks to microseconds* *static long Sec2Tick(double sec);* *//converts seconds to ticks* *static long MSec2Tick(double msec);* *//converts milliseconds to ticks* *static long USec2Tick(double usec);* *//converts microseconds to ticks* *static double getSec(long ticks);* *//returns current time in seconds* *static double getMSec(long ticks);* *//returns current time in milliseconds* *static double getUSec(long ticks);* *//returns current time in microseconds*
SimEvent.java	Every SimComponent communicates with each other by enqueuing SimEvent for the target component. For example, when component A wants to send a packet to component B, component A creates a SimEvent that specifies B as its destination and enqueue the event. The SimEvent object also contains a time so that this is fired at exactly the specified time. Component B will then be able to react to the event accordingly.

University of Malaya

The constructor for SimEvent class is shown below:

```
SimEvent(int aType, simComponent src, SimComponent dest,
        long aTick, Object[] params);
   /*the constructor needs an event type (as defined in
   SimProvider or a private event type), the source and
   destination SimComponent, a time (in ticks) and an array of
   java.lang.Object (which can be anything) holding various
   parameters for the event. The event type determine what
   the array will contain*/
```

Upon receiving the SimEvent object, its content can be retrieved the following functions:

```
int getType();
   //gets the event type
SimComponent getSource();
   //gets the source SimComponent
SimComponent getDest();
   //gets the destination SimComponent
long getTick();
   //gets the time (in ticks) to fire the events
Object[] getParams();
   //gets the event's parameters
```

University of Malaya

SimComponent.java	SimComponent class is the most important class in the simulator in order to develop new components. Network components like GenericATMSwitch, GenericLink, GenericBTE, CBRApp and VBRApp inherit from SimComponent. This class provides the skeleton for an actual component. A new component should extend SimComponent and override its various methods in order to provide meaningful operations for the component. The constructor for SimComponent class is shown below: *mComponent(String aName, int aClass, int aType, JavaSim aSim, Point loc);* */*every new component must provide a constructor with exactly the above parameters, super(aName, aClass, aType, aSim, loc) function is immediately called as the first statement of the method in order to override its parameters*/* The SimComponent also provide a set of functions according to its types of operations. Among the operations are neighboring operation, copy operation, initial or reset operation and event handler operation. The functions in neighboring operation are *addNeighbor, removeNeighbor,*

University of Malaya

removeNeighbors and isConnectable. Any component that needs to handle neighbor connect or disconnect operations should override these methods.

void addNeighbor(SimComponent comp);

*/*the simulation engine calls this function when a new neighbor is connected to this component*/*

void removeNeighbor(SimComponent comp);

*/*the simulation engine calls this function when a neighbor is disconnected from this component*/*

void removeNeighbors(java.util.List comp)

*/*the simulation engine calls this function when a group of neighbors is disconnected from this component*/*

boolean isConnectable(SimComponent comp)

*/*the simulation engine calls this function when a new component is about to be connected to this component. This function checks whether two component can be connected together*/*

*/*this connection rules are:*

1. *Application (CBRApp or VBRApp) only allowed connecting to BTE.*
2. *BTE only allowed to connect to Application (CBRApp or VBRApp) and GenericLink*
3. *GenericLink only allowed to connect to GenericBTE and GenericATMSwitch*

University of Malaya

4. GenericATMSwitch only allowed to connect to GenericLink/*

The only function in copy operation is copy.

```
void copy(SimComponent comp);
  /*this method is used to copy parameter values of another
  SimComponent of the same type. This method must be
  override in order to ensure that all necessary parameter
  values are copied.*/
```

The functions in initial or reset operation are reset, start and resume.

```
void reset();
  /*this function brings the status of the component back to
  the same status as if it is just newly created.*/
void start();
  /*this function starts the simulation when the user click the
  "Start" button*/
void resume();
  /*one possible use of this function is to capture any special
  changes that have been done by the user during the pause
  period. It takes action when user clicks the "Resume" button
  after a pause.*/
```

University of Malaya

	The function in event handler operation is action. *void action(SimEvent e);* */*this is the event handler of this component and will be called by the simulator engine whenever a SimEvent with this component as the destination fires.*/*
SimParameter.java	Classes like SimParamInt, SimParamDouble, SimParamBool, SimParamNSAP and SimParamRTable inherit from SimParameter. These classes provide support for integer, double and boolean parameters. Extending SimParameter accordingly can create other types of parameters. By extending SimParameter, these classes can obtain parameter logging and meter display features automatically. The constructor for SimParameter class is shown below: *nParameter(String aName, String compName, long creationTick, boolean isLoggable);* */*the parameters for SimParameter constructor are:* *1. aName – name of the parameter* *2. compName – name of the component the owns that parameter* *3. creationTick – time when the parameter is created*

University of Malaya

	4. *isLoggable – whether the parameter can be logged in the log file*

Table 4.1: Major functions for java files including in UMJaNetSim

4.4 Chapter Summary

In this chapter, UMJaNetSim had been described in details for the architecture, function, classes and methods. By understanding more on the UMJaNetSim, it is good for the following stage which is to develop the Traffic Conditioner into UMJaNetSim.

University of Malaya

CHAPTER 5: DESIGN

5.1 System Design

In this phase, parameters required for traffic conditioning will be design in order to let user to input sufficient information and the plans of traffic conditioning are transformed into reality. Figure 5.1 show traffic flow handling stage in traffic conditioning which include classifier, meter, marker, shaper and dropper. After showing traffic flow of traffic conditioning using Token Bucket and Leaky Bucket respectively, the design of simulator which including traffic conditioning is converted into designing the coding. Object oriented methodology is used at the implementation phase of the development process through Java programming language.

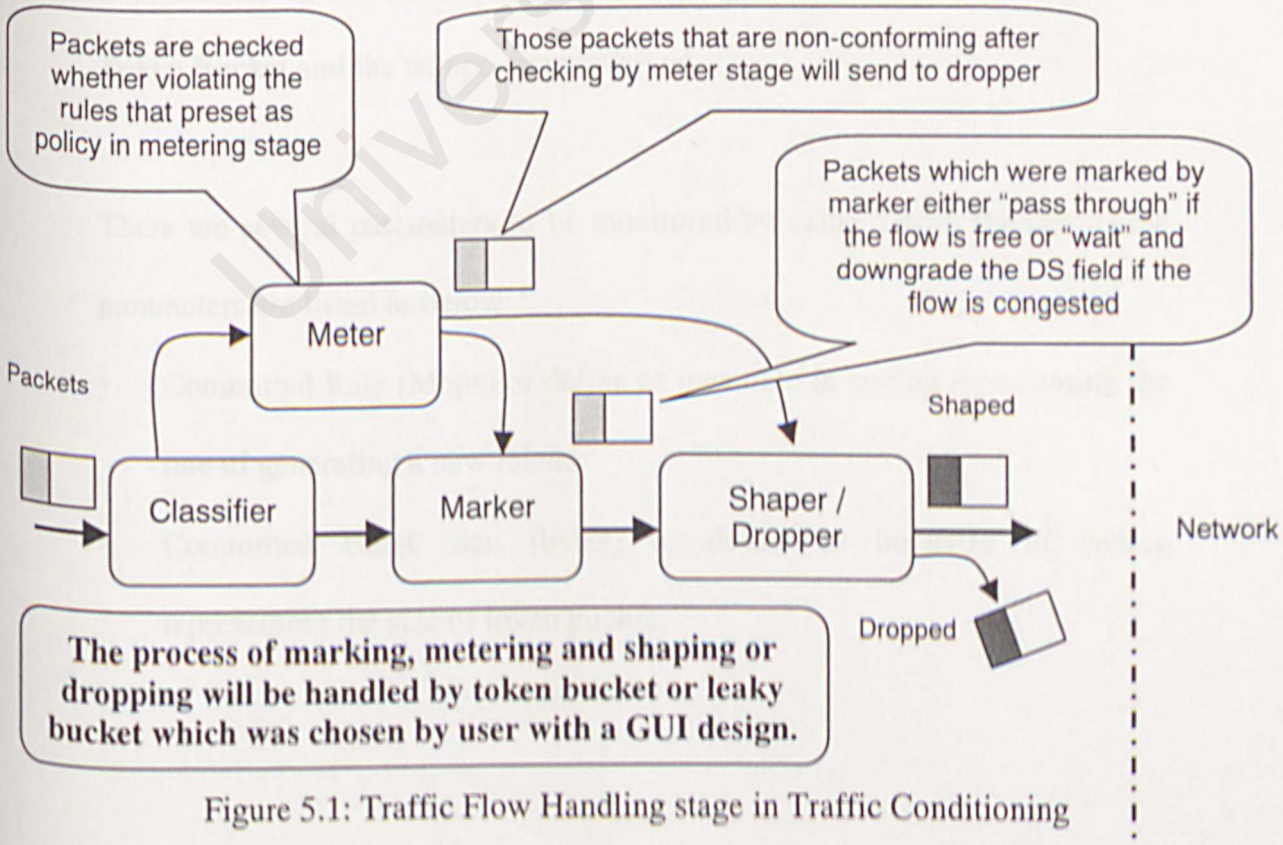


Figure 5.1: Traffic Flow Handling stage in Traffic Conditioning

University of Malaya

In order to implement a traffic conditioner into JaNetSim, there are several parameters to be configured which are Peak Rate, Committed Rate and Committed Burst Size. These three parameters will be considered as profile of Traffic Conditioner.

5.1.1 Token Bucket Design

A Token Generator, Token Bucket and Data Buffer and data buffer is required in order to implement Token Bucket technique in the router. Token generator is designed to generate new token in the time interval that has been specified. Token Bucket is used to keep generated token. If token bucket is full of token at the moment, the new generated token will be discarded. Meanwhile, data buffer is function to queue the incoming cells in a buffer before getting a token and admit to network. Both the Token Generator and Token Buffer will be implemented at the input port of router. Figure 5.2 show the designed of Token Bucket and the components involved.

There are several parameters to be monitored by using Token Bucket. These parameters are listed as below:

- Committed Rate (Mbps) or define as meanrate in coding representing the rate of generating a new token.
- Committed Burst Size (bytes) or define as burstsize in coding representing the size of token bucket.

University of Malaya

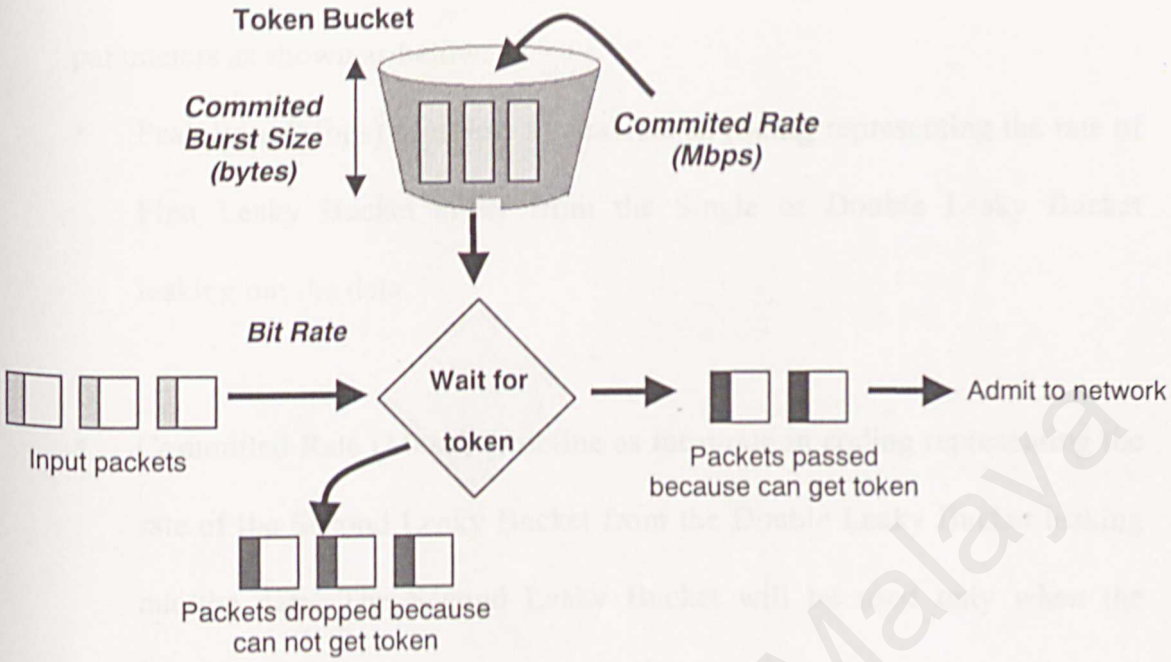


Figure 5.2: Token Bucket and the components involved

5.1.2 Leaky Bucket Design

Leaky Bucket is a data buffer where if the incoming data is allow by the policer, then the data can admitted into network, else the data will be dropped. The policer is the parameters that required monitoring the traffic. There are two types of parameters which are the leaking rate of the bucket and the size of bucket.

The Leaky Bucket is designed in a flexible way. In every input port, there will be two types of Leaky Bucket:

- Single Leaky Bucket and
- Double Leaky Bucket

University of Malaya

In order to use Leaky Bucket for traffic policing, user require setting some parameters as shown as below:

- Peak Rate (Mbps) or define as peakrate in coding representing the rate of First Leaky Bucket either from the Single or Double Leaky Bucket leaking out the data.

- Committed Rate (Mbps) or define as meanrate in coding representing the rate of the Second Leaky Bucket from the Double Leaky Bucket leaking out the data. The Second Leaky Bucket will be used only when the Committed Rate set.

- Committed Burst Size (bytes) or define as burstsize representing the size of Second Leaky Bucket from the Double Leaky Bucket.

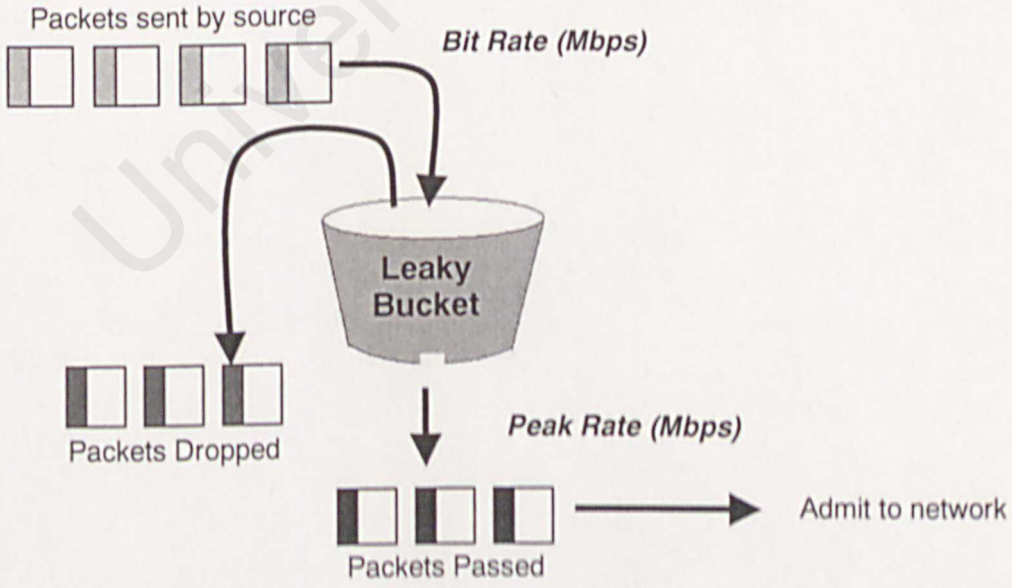


Figure 5.3: Design of Single Leaky Bucket

University of Malaya

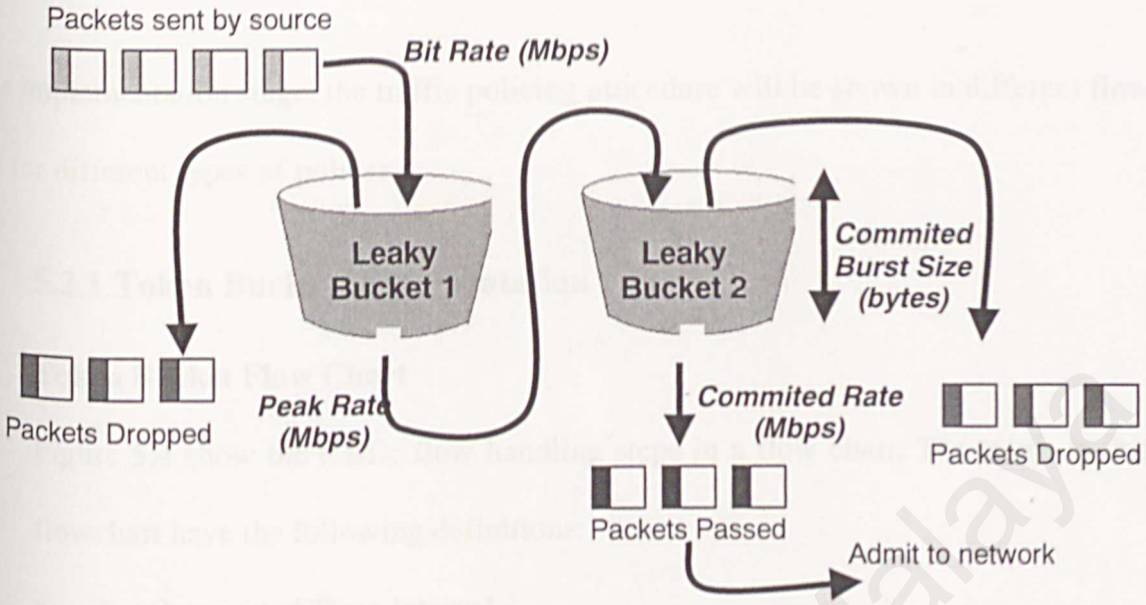


Figure 5.4: Design of Double Leaky Bucket

University of Malaya

5.2 System Implementation

In the implementation stage, the traffic policing procedure will be shown in different flow chart for different types of policer.

5.2.1 Token Bucket Implementation

Token Bucket Flow Chart

Figure 5.4 show the traffic flow handling steps in a flow chart. The terms in this flowchart have the following definitions:

- Tc: Committed Time Interval
- FIFO: first-in-first-out

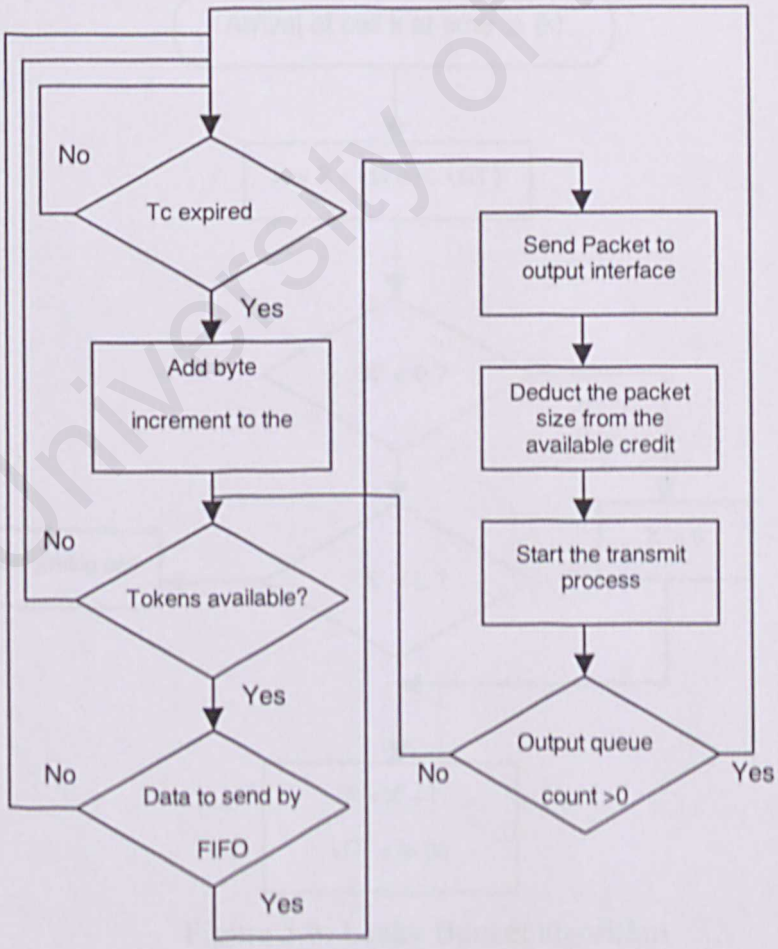


Figure 5.5: Flow Chart of Token Bucket Algorithm

University of Malaya

5.2.2 Leaky Bucket Implementation

Leaky Bucket Flow Chart

This is known as continuous state for general leaky bucket algorithm. Definitions of the variables used in the Figure 5.5 are:

- I = Increment
- L = Limit
- ta (k) = Time of arrival of a cell
- X = Value of the leaky bucket counter
- X' = Auxiliary variable
- LCT = Last compliance time

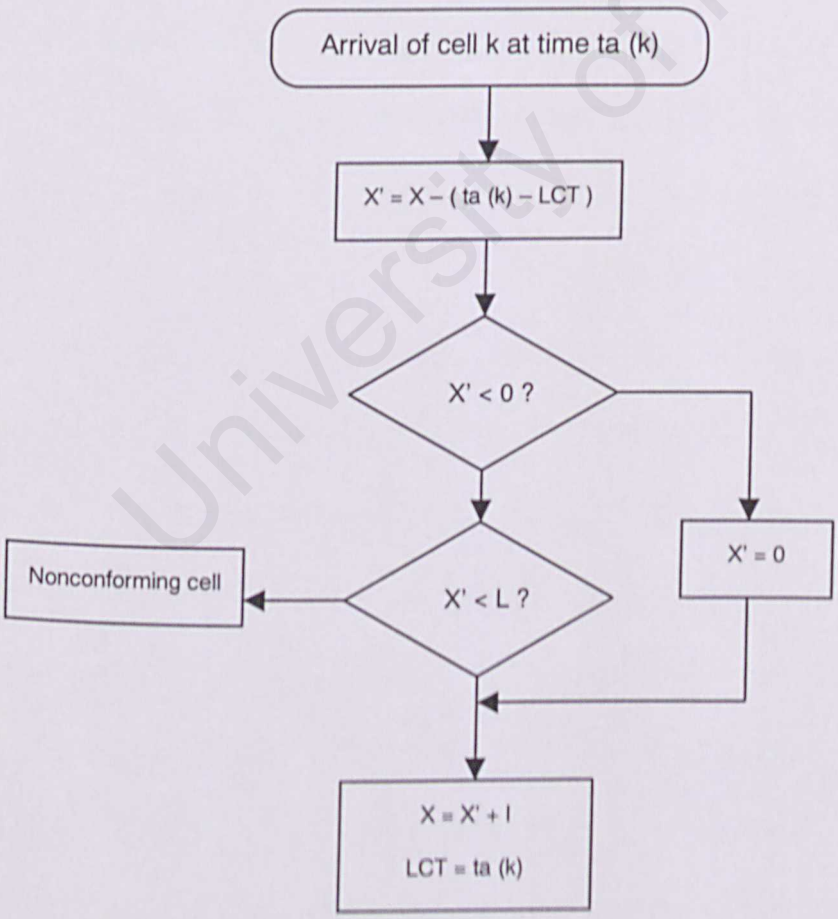


Figure 5.6: Leaky Bucket algorithm

University of Malaya

5.3 Chapter Summary

This chapter had discussed the design for the work flow for Traffic Conditioner including Leaky Bucket and Token Bucket. The following chapter will apply the approaches stated in this chapter to implement a Traffic Conditioner to police the network traffic in order to avoid network congestion.

University of Malaya

CHAPTER 6: IMPLEMENTATION

In the implementation phase, the plans in design stage are transformed into reality. The designed simulator is converted into coding. The implementation approach used in Traffic Conditioning By Using Leaky Bucket and Token Bucket are the object modeling. Object oriented methodology is used at the implementation phase of the development process through the Java programming language.

6.1 System Implementation

Traffic Conditioning is implemented into router (IPRouter.java) in JaNetSim and this function is implemented to test on CBR application for UDP (UDP_CBR.java). Therefore, some methods are added into both IPRouter.java and UDP_CBR.java.

Besides that, the parameters involving in traffic conditioning are Peak Rate, Committed Rate and Committed Burst Size. These three parameters will be written in a TrafficProfile.java. and will be explained later.

By sending data into network, all packets will be converted into frames (multiple sizes controlled by user) before policing by traffic conditioner. In order to apply traffic conditioner for policing the data, some methods are added into IPPacket.java and EtherFrame.java.

University of Malaya

6.2 Implementation of Traffic Conditioning

The following algorithms had been used to develop Traffic Conditioner in JaNetSim.

6.2.1 Leaky Bucket

If the user enable the traffic conditioning function and choose Leaky Bucket as a policer to police the traffic, the following coding will be execute in order to define the value which are useful for Leaky Bucket in the class named tc_performed( ).

```
//leaky bucket 1: meter peak rate
prof.LB1_TAT = theSim.now( );
prof.LB1_I = SimClock.USec2Tick ( frame.len * 8 / frame.t_profile.peakrate );
prof.LB1_L = prof.LB1_I

//leaky bucket 2: meter mean rate
prof.LB2_TAT = theSim.now( );
prof.LB2_I = SimClock.USec2Tick ( frame.len * 8 / frame.t_profile.meanrate );
prof.LB2_L = ( long ) ( ( frame.t_profile.burstsize / frame.len -1) * (prof.LB2_I –
             prof.LB1_I ) + prof.LB1_I );
```

As shown in the coding below, prof.LB1_TAT, prof.LB1_I and prof.LB1_L are variable useful to police for the Single Leaky Bucket and first leaky bucket for Double Leaky Bucket. Meanwhile prof.LB2_TAT, prof.LB2_I and prof.LB2_L are variable useful for second leaky bucket for Double Leaky Bucket. Three parameters from the profile (peakrate, meanrate, burstsize) are used to define the value useful for bucket. Besides that, theSim.now( ) representing the time for current tick and the frame.len in the coding representing the packet size that send by application to router.

University of Malaya

In the class of checkConforming( ), the following coding indicates that if the sw_policer is 0, then user is choosing Leaky Bucket as a traffic policer.

```
switch(sw_policer.getValue()) {
case 0: //leaky bucket
    //first bucket
    if(now>prof.LB1_TAT) {
      prof.LB1_TAT=now;
    }
    else if(now+prof.LB1_L < prof.LB1_TAT) {
          return -2;
    }
    prof.LB1_TAT+=prof.LB1_I;

    //second bucket
    if(prof.LB2_L>=0){
        if(now>prof.LB2_TAT) {
         prof.LB2_TAT=now;
        }
        else if(now+prof.LB2_L < prof.LB2_TAT)
  {
          return -2;
  }
  prof.LB2_TAT+=prof.LB2_I;
        }

    break;
}
return 0;
```

As shown in coding above, the system is comparing the available time tick in the first and second bucket for transferring the frames. If there are no available time ticks, the frame will be dropped and the method will return -2, else the frame will be admitted into network and the method will return default value, 0.

If the user decides to use Single Leaky Bucket, only Peak Rate (peakrate)is required to set for the value. On the other hand, if the user set the value for Committed Rate (meanrate) and Committed Burst Size (burstsize) means the user decides to use Double Leaky Bucket.

University of Malaya

6.2.2 Token Bucket

On the other hand, if the user enable the Traffic Conditioning and choose Token Bucket as a policer to monitor the traffic, the following coding will be executed in order to define the values which are useful for Token Bucket in the class of tc_performed( ).

```
//token bucket
prof.token_count1=frame.t_profile.burstsize;
prof.lastpackettime=theSim.now( );
```

As shown in the coding above, burstsize is used to define prof.token_count1 and the time for current tick or theSim.now( ) is used to define prof.lastpackettime. After defining the useful value, it will be used in class named checkConforming( ) which will be explain in the following paragraph.

In the class of checkConforming( ), the following coding indicates that if the sw_policer is 1, then user is choosing Token Bucket as a traffic policer. The following coding indicates that the system will generate token using Commited Rate or known as meanrate and token size is defined as bytes. If there are available token for frames (must be equal or more than a packet size), then the frame will be admit to the network and the method will return 0, then the system will update the available tokens in bucket. On the other hand, the frame will be dropped and the method will return -2 if there are no available token.

University of Malaya

```
switch(sw_policer.getValue()) {
case 1: //token bucket

    bytes = ( int ) ( frame.t_profile.meanrate * SimClock.Tick2Usec ( now –
            prof.lastpackettime ) / 8 );

    if( bytes > 0 && ( prof.token_count1 < frame.t_profile.burstsize ) ) {
       int diff = frame.t_profile.burstsize - prof.token_count1;

       if(diff >= bytes) {
           prof.token_count1 += bytes;
           bytes = 0;
        }
        else {
           prof.token_count1 = frame.t_profile.burstsize;
           bytes -= diff;
        }
       }
       prof.lastpackettime = now;

       if( (prof.token_count1 - frame.len) >= 0 ) {
           prof.token_count1 -= frame.len;
           return 0;
       }
       return -2;
   }
   return 0;
}
```

University of Malaya

6.3 Implementation methods

In the following part, the modification for the java files and the added java files will be explained. However, the following part only explaining the important methods that had been modified and the minor modification will be printed and attached at Appendix.

6.3.1 IPRouter.java

The following three methods are added into IPRouter.java:

```
protected boolean tc_perform(EtherFrame frame,Port voport)
{       /////perform traffic conditioning here/////

        /*if the profile for traffic conditioning is not null, then define each of the value
        for the parameters required to police the traffic.*/

        /*pass the parameters define by the traffic profile to the methods of
        checkConforming( frame, prof ). If the returns result less than -1, the frames
        will be dropped, else the frames will be transmitted to network.)*/

        /*if the action is to drop the frames, then num_dropped will increase 1 and
        calculate the percentage of drop and pass frames overall.*/

        /*notify the application about the quantity of frames dropped after pass
        through the traffic conditioner.*/
}
private int checkConforming (EtherFrame frame,ProfileRecord prof)
{
        //main purpose is to test whether a frame conforms to its traffic profile

        /*check whether the user choose to use Leaky Bucket or Token Bucket to
        police the traffic*/

        /*return either 0 or -2 to tc_perform ( EtherFrame, Port) where 0 if strictly
        conforming and -2 if the frames violate the profile.*/
}

protected void sw_averaging_interval()
{
  /*calculate the percentage of frames dropped by the policer.*/
}
```

University of Malaya

Apart from that, there are some minor modifications which are stated as below:

```
if ( sw_tc.getValue() = = true && packet.t_profile!= null )
{
    t_profile=packet.t_profile;
}
```

From the coding above, the system will do comparison, if the user ticks the check box for "Enable Traffic Profile" in router's property and the profile for the coming packet from application is not null, then system will assign the current router's profile for traffic conditioning following the profile of the coming packet.

```
.update( theSim.now( ) )
```

The coding above is to update the changing value to SimComponent.java in order to draw a graph to show the value at on moment.

```
frame.sender=packet.sender;
```

Every frame and packet has a sender in order to transfer relevant information. Form the coding above, frame.sender is assign and equal to packet.sender in order to notify to the application about the total frames dropped.

6.3.2 UDP_CBR.java

There following modification had been done to UDP_CBR.java

```
public Object compInfo ( int infoid, SimComponent src, Object paramlist )
{
        /*if the frame violating the profile of traffic conditioner, then the frame will be
        dropped at router and application received the information and update the
        number of frames drop here*/
}
```

University of Malaya

From the coding below, if the user ticks the check box for "Enable Self Policing" in application's property, the system will define a new profile to the application and define the value set by user as the profile of the system.

```
if(cn_tc.getValue()==true)
{
  t_profile=new TrafficProfile();
  t_profile.peakrate=cn_peakrate.getValue();
  t_profile.meanrate=cn_meanrate.getValue();
  t_profile.burstsize=cn_burstsize.getValue();
  t_profile.burstsize2=cn_burstsize2.getValue();
}
```

6.3.3 TrafficProfile.java

TrafficProfile.java is added into JaNetSim in order to declare the parameters of traffic conditioner for as traffic profile.

```
package janetsim.component;
class TrafficProfile implements java.io.Serializable
{
        // declare Peak Rate
        //declare Committed Rate
        //declare Committed Burst Size
}
```

6.3.4 IPPacket.java / EtherFrame.java

Every packet in order to transmit to network must have it's relevant information.

```
TrafficProfile t_profile=null;
```

The coding above creating a new empty TrafficProfile called t_profile for every different packet in order to define the parameters later.

University of Malaya

```
SimComponent sender=null;
long sendingTick;
```

This coding is to create a SimComponent called sender and assign it to null. Then declare another variable sendingTick as long.

```
IPPacket( ){ };
```

From the coding above, the system will build an empty contructor for IPPacket.

```
IPPacket(SimComponent thesender,long tick)
{
  //define the value of sender;
  //define the value of sendingTick;
  }
```

6.4 Chapter Summary

This chapter has discussed the system implementation phase in the Traffic Conditioner. The conceptual and abstract designs were transformed into programming languages codes. The implementation of two Traffic Conditioner techniques – Leaky Bucket and Token Bucket are described in detail. Further more, the details implementation for the proposed algorithms also included.

University of Malaya

CHAPTER 7: TESTING

7.1 Unit and Class Testing

Unit testing section is separated into two parts. The first part will carry out the Leaky Bucket unit testing while the second part will carry out the Token Bucket unit testing.

7.1.1 Leaky Bucket Unit and Class Testing

Leaky Bucket is divided into Single Leaky Bucket and Double Leaky Bucket. Unit testing that done for the Leaky Bucket using the parameters set in the bucket to police the traffic. There are few profiles which are needed to police the traffic by using Leaky Bucket. These profiles are:

- Peak Rate (Mbps) - peakrate
- Committed Rate (Mbps) - meanrate
- Committed Burst Size (Mbps) - burstsize

These three parameters had been used to define few variable in the class of tc_perform( ).

For Single Leaky Bucket, unit testing is done using the following example of value where each packet is defined as 53 bytes by user and the SimClock.Usec2Tick( ) is to divided the parameter by 0.001.

```
//leaky bucket 1: meter peak rate
prof.LB1_TAT = theSim.now( );
prof.LB1_I = SimClock.USec2Tick ( frame.len * 8 / frame.t_profile.peakrate );
prof.LB1_L = prof.LB1_I;
```

University of Malaya

Input	Output
peakrate = 11	prof.LB1_TAT = 527
	prof.LB1_I = 0.038545454545454545454545454545455
	prof.LB1_L = 0.038545454545454545454545454545455

Table 7.1: Output of Single Leaky Bucket

If the prof.LB1_TAT, prof.LB1_I and prof.LB1_L are having the same value as shown as the output mean that the class tc_perform( ) is successfully implemented. Then these three output values will be used to compare the time tick in checkConforming ( ) for Single Leaky Bucket to check whether the frame is allow to admit to the network or not.

For Double Leaky Bucket, unit testing is done using the following example:

```
//leaky bucket 2: meter mean rate
prof.LB2_TAT = theSim.now( );
prof.LB2_I =SimClock.USec2Tick ( frame.len * 8 / frame.t_profile.meanrate );
prof.LB2_L = ( long ) ( ( frame.t_profile.burstsize / frame.len – 1 ) * ( prof.LB2_I -
             prof.LB1_I ) + prof.LB1_I );
```

Input for first bucket	Output for first bucket
peakrate = 11	prof.LB1_TAT = 527
	prof.LB1_I = 0.038545454545454545454545454545455
	prof.LB1_L = 0.038545454545454545454545454545455

Table 7.2: Output of first leaky bucket of Double Leaky Bucket

Input for second bucket	Output for second bucket
meanrate = 10	prof.LB2_TAT = 580
burstsize = 53	prof.LB2_I = 0.0424
	prof.LB2_L = 0.038545454545454545454545454545455

Table 7.3: Output of second leaky bucket of Double Leaky Bucket

If the prof.LB1_TAT, prof.LB1_I prof.LB1_L, prof.LB1_TAT, prof.LB1_I and prof.LB1_L are having the same value as shown as the output mean that the class tc_perform( ) is successfully implemented. These six parameters will be used to compare the time tick in checkConforming ( ) for Double Leaky Bucket to check whether the frame is allow to admit to the network or not.

Leaky Bucket is a discrete event case where the variables are kept on changing and it is hard to predict the value for variables at a particular time. However, theoretically for both Single and Double Leaky Bucket, the variable defined in tc_perform( ) will be used in checkConforming as shown at the coding below to police the traffic. The following paragraph will explain how to prove that the Leaky Bucket is successfully running.

```
//first bucket
    if( now > prof.LB1_TAT ) {
       prof.LB1_TAT = now;
    }
    else if( now + prof.LB1_L < prof.LB1_TAT ) {
          return -2;
    }
    prof.LB1_TAT += prof.LB1_I;

//second bucket
    if( prof.LB2_L >= 0 ) {
          if( now > prof.LB2_TAT ) {
                prof.LB2_TAT = now;
          }
          ese if( now + prof.LB2_L < prof.LB2_TAT ) {
                return -2;
          }
          prof.LB2_TAT += prof.LB2_I;
    }

    break;
}
return 0;
```

When a quantity of frames admitted to a router, the new arrival time (theSim.now( ))for each frame will be updated. Then prof.LB1_TAT must be more than or equal to theSim.now( ). If the value of theSim.now ( ) + prof.LB1_L is less than prof.LB1_TAT, return result -2 , then increasing number of frame dropped with 1 and drop the particular frame. On the other hand, if the value of theSim.now ( ) + prof.LB1_L is more than or equal to prof.LB1_TAT, return result 0, no frames is dropping, increasing the number

of frame passed with 1 and admit the frame into network. The same thing happens to the second bucket in Double Leaky Bucket. From this test, it proved that the Single and Double Leaky Bucket are working properly and successfully.

7.1.2 Token Bucket Unit and Class Testing

In the Token Bucket Testing stage, there are two parameters from the fused to define the variable useful for policing the network traffic. There are:

- Committed Rate (Mbps) - meanrate
- Committed Burst Size (Mbps) – burstsize

Committed Rate is used to define the variable called meanrate and Committed Burst Size is used to define the variable called burstsize (prof.token_count1).

In order to test the Token Bucket, there are several aspects to be considered. First of all, if user specifies the time to generate a token in router is 10 Usec. By setting the value 10 Mbps for the Committed Rate or meanrate, the token is generated at 10 Mbits in a second which is equal to generate a token for every 10 Usec.

Secondly, the Commited Burst Size is specified to 530. When the generated token is put into the bucket, the bytes (generated token) is increased. No further increment is done when the bucket counter reached the value of Committed Burst Size which is 530. This proved that the Token Bucket could keep the generated token until the maximum value or burstsize

University of Malaya

(prof.token_count1). On the other hand, when a quantity of frames are admitted to the network passed by policer, the bucket counter will be decreased. It means that the cells successfully get the token from the Token Bucket.

Thirdly, all the frames will be policed when it reaches to router. After check for available token, the frame will either admit to network or dropped. If there are no available token, the methods will return result -2, else the methods will return 0 in order to admit to network. If all the three aspect stated above is correct, it proved that the Token Bucket is implemented successfully.

University of Malaya

7.2 System Testing

System Testing is separated into two major parts which are Leaky Bucket Testing and Token Bucket Testing. Moreover, the Leaky Bucket Testing is further divided into Single Leaky Bucket Testing and Double Leaky Bucket Testing.

A simple topology is created for the system testing by using different parameters. The topology consists of two routers (called Router 1 and Router 2), an UDP CBR application (called App 1) and a link (called Link 1). The created topology is shown in figure below:

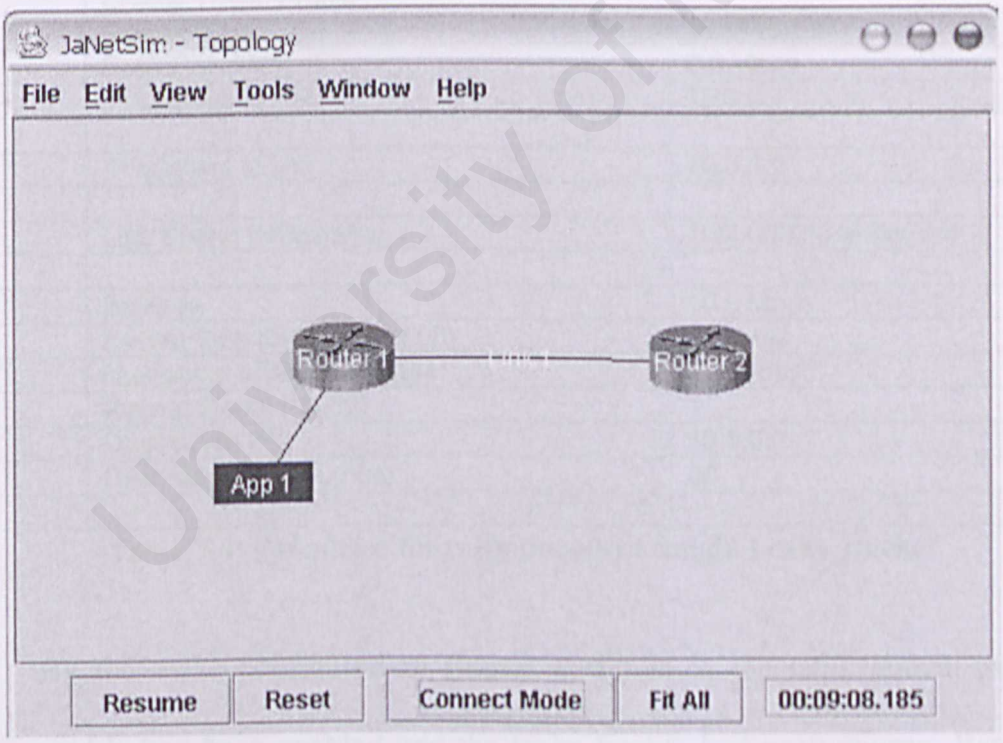


Figure 7.1: Simple Topology to test Leaky Bucket and Token Bucket

University of Malaya

7.2.1 Leaky Bucket System Testing

a) Single Leaky Bucket

By testing the functionality of Single Leaky Bucket, the topology shown in Figure 7.1 is used. App 1 is configured to send packet to the destination router, Router 2 pass through Router 1 by using Link 1. Traffic Conditioner is enable in Router and the profile of Traffic Conditioner is set at Router to police the network traffic. The parameters entered in the corresponding routers, application and link are shown in Table 7.4. If the parameters not mention in the following table mean it retain the default value.

Components	Parameters	Value
Router 1	Enable Traffic Profile	√
	Policer Type	leaky bucket
	Peak Rate (Mbps)	9.0 Mbps
	IP address to link	10.0.0.1
Router 2	IP address to link	10.0.0.2
Link 1	Link Speed (MBits/sec)	10000000.0 MBits/sec
App 1	Bit Rate	10.0 Mbps
	Packet Size (bytes, 46-1500)	53 bytes
	Number of MBits to be sent	1.0 MBits
	Repeat Count (-1=inf)	1
	Destination IP	10.0.0.2
	Destination port number	80

Table 7.4: Parameter for components of Single Leaky Bucket

By the value configured in Router 1 shown in the table above, traffic profile had been enabled and the leaky bucket had been chosen. Besides that, only the Peak Rate is defined as 9.0 Mbps which indicating Single Leaky Bucket is applied for Traffic Conditioner. The IP address for Router 1 is 10.0.0.1. Meanwhile, IP address for Router 2 must be under the same network with Router 1 in order to communicate, therefore, it set to be

10.0.0.2. Then the Link Speed for the link must be set to a large number bigger than the Bit Rate of the application in order to make not congested because of Bit Rate violating the Link Speed, therefore we assume that no link congestion in our test cases. For App 1, the Bit Rate is set to 10 Mbps and the packet size is 53 bytes, Number of MBits to be sent is 1.0 MBits, Repeat Count is 1, Destination IP is the 10.0.0.2 and the Destination port number is 80.

Bit Rate at the App 1 representing the sending rate of the packets for the application. For Single Leaky Bucket, only the Bit Rate is policing and some of the frames will be dropped if the Bit Rate violating the Peak Rate set at Router. From the table above, Bit Rate 10 Mbps is violated the Peak Rate 9.0 Mbps which indicating there App 1 violating 10% of the profile for Router 1 and there are roughly 10% of frames dropped at Router 1 and the total number of frames dropped will be shown at the property for App 1. The rest 90% of frames will be admitted into network and send to destination router, Router 2 with IP address 10.0.0.2 by using port 80.

Result for this test case is shown at the table below:

Components	Parameters	Value
Router 1	Frame Drop %	9.9618482407799%
	Frames Passed By TC	2122
	Frames Dropped By TC	235
Router 2	Frames Received	2124
App 1	Total frames sent	2358
	Total frames dropped by tc	235

Table 7.5: Result for components of Single Leaky Bucket

b) Double Leaky Bucket

By testing the functionality of Double Leaky Bucket, the same topology is used but some of the parameters entered in the corresponding routers, application and link are different and shown in Table 7.6. If the parameters not mention in the following table mean it retain the default value.

Components	Parameters	Value
Router 1	Enable Traffic Profile	√
	Policer Type	leaky bucket
	Peak Rate (Mbps)	10.0 Mbps
	Committed Rate (Mbps)	8.0 Mbps
	Committed Burst Size (bytes)	53 bytes
	IP address to link	10.0.0.1
Router 2	IP address to link	10.0.0.2
Link 1	Link Speed (MBits/sec)	10000000.0 MBits/sec
App 1	Bit Rate	10.0 Mbps
	Packet Size (bytes, 46-1500)	53 bytes
	Number of MBits to be sent	1.0 MBits
	Repeat Count (-1=inf)	1
	Destination IP	10.0.0.2
	Destination port number	80

Table 7.6: Parameter for components of Double Leaky Bucket

Most of the value is configured following the Single Leaky Bucket but Peak Rate had been changed to 10 Mbps, the Committed Rate is set to 8.0 Mbps and Committed Burst Size is set to 53 bytes for Router 1. By defining the value for Committed Rate and Committed Burst Size, Double Leaky Bucket is chosen.

For this case, Bit Rate 10.0 Mbps at the App 1 is no violating the Peak Rate also 10.0 Mbps or not violating the rules for first leaky bucket. When it reach the second leaky bucket, Bit Rate is now violating the Committed

Rate 8.0 Mbps although there are enough burst size provided (mean the second bucket able to fetch 53 bytes of frame in on time). The App 1 violating 20% of the profile for Router 1 and there are roughly 20% of frames dropped at Router 1 and the total number of frames dropped will be shown at the property for App 1. The rest 80% of frames will be admitted into network and send to destination router, Router 2 with IP address 10.0.0.2 by using port 80.

Result for this test case is shown at the table below:

Components	Parameters	Value
Router 1	Frame Drop %	19.966087325137%
	Frames Passed By TC	1886
	Frames Dropped By TC	471
Router 2	Frames Received	1888
App 1	Total frames sent	2358
	Total frames dropped by tc	471

Table 7.7: Result for components of Double Leaky Bucket

University of Malaya

7.2.2 Token Bucket System Testing

By testing the functionality of Token Bucket, the same topology is used but some of the parameters entered in the corresponding routers, application and link are different compare to the Leaky Bucket parameters and this will shown in Table 7.5. If the parameters not mention in the following table mean it retain the default value.

Components	Parameters	Value
Router 1	Enable Traffic Profile	√
	Policer Type	Token bucket
	Committed Rate (Mbps)	9.0 Mbps
	Committed Burst Size (bytes)	53 bytes
	IP address to link	10.0.0.1
Router 2	IP address to link	10.0.0.2
Link 1	Link Speed (MBits/sec)	10000000.0 MBits/sec
App 1	Bit Rate	10.0 Mbps
	Packet Size (bytes, 46-1500)	53 bytes
	Number of MBits to be sent	1.0 MBits
	Repeat Count (-1=inf)	1
	Destination IP	10.0.0.2
	Destination port number	80

Table 7.8: Parameter for components of Token Bucket

In order to test for Token Bucket, the parameters had been set as shown at the table above.

For this case, Bit Rate 10.0 Mbps at the App 1 violating the Committed Rate 9.0 Mbps although there are enough burst size provided (mean the token bucket having 53 bytes of available token to fetch a frame in on time). For Token Bucket, the App 1 violating 50% of the profile for Router 1 and there are roughly 50% of frames dropped at Router 1 and the total number of frames dropped will be shown at the property for App 1. The Router dropped 50% of

University of Malaya

frames because when the first frames from App 1 reach to Router 1, it get 53 bytes of token from the bucket. However, the second frame sends faster than the rate of generating the token where the second frame will be dropped. Then the third frame will be the token but the fourth frame can not get the token. From this test case, among two frames, there are one frame will be dropped which indicate 50% of frames will be dropped totally. The rest 50% of frames will be admitted into network and send to destination router, Router 2 with IP address 10.0.0.2 by using port 80.

Result for this test case is shown at the table below:

Components	Parameters	Value
Router 1	Frame Drop %	49.93641373463332%
	Frames Passed By TC	1179
	Frames Dropped By TC	1178
Router 2	Frames Received	1181
App 1	Total frames sent	2358
	Total frames dropped by tc	1178

Table 7.9: Result for components of Token Bucket

Apart from reading the result showing in the label, user can view the result by showing it into a graph. By ticking the check box located left for the label, user can view the result by graph.

University of Malaya

7.3 Chapter Summary

This chapter has discussed the testing process in detail. The components are tested for their functionality and behavior during different operations. Testing is done for Single Leaky Bucket, Double Leaky Bucket and finally Token Bucket. The test has proved that the simulator is successfully executed with the designed model. Further more, there are some others test case with result shown in Appendix.

University of Malaya

CHAPTER 8: CONCLUSION

8.1 Project Finding

From this project, there are different kinds of experience of computer science are gained. By study more detail for the existing JaNetSim version 0.66 and JaNetSim version 0.49, I know more about network simulator. Besides that, the most important thing is the understanding of Traffic Conditioning such as Leaky Bucket and Token Bucket using for network simulator. Moreover, the experience gained in problems solving while designing, implementing and testing of the simulator is invaluable.

In order to develop a good and efficient simulator, choosing a suitable language is very important. Therefore Java Programming Language which is object-oriented programming language had fulfilled and satisfied all the conditions needed to construct the simulator. It has features such as reusability, maintainability, extensibility and modifiability. It also enables the developed simulator support multithreading and the ability of the threads to run simultaneously.

University of Malaya

8.2 Objectives Achieved

The objectives for this project have been achieved and they are explained clearly in previous chapters.

A through survey has made on some current network simulators in order to gain better insight into the working of a network simulator. Special attention is given to the methods currently being used to overcome congestion in a network environment. By implementing traffic conditioner in JaNetSim version 0.66 which is used as a congestion controller is this project is very efficient. Due to this reason, Leaky Bucket and Token Bucket techniques are considered to be ideal techniques in controlling the traffic in a network.

The objective of this project is to implement Leaky Bucket and Token Bucket in JaNetSim version 0.66. With creativity and innovations, the various components in Leaky Bucket and Token Bucket are successfully written and tested to produce the expected results.

University of Malaya

8.3 Simulator Strength

The developed UMJaNetSim has its strengths. These strength are listed below:

- **Platform independent**

  The developed simulator using Java Programming Language enables it to be cross platform use purpose. Thus, the simulator works well in Windows, Unix and Linux environment.

- **Simple and User Friendly Interface**

  The designed graphical user interface facilities user in creating a network topology on the workspace. The menu items and components in the Control Bar provide the essential function for user to create, save, open or modify the network topology easily. Creation of a network component is just a click on the workspace interface. User can enter or modify parameters in each component in the pop up properties box.

- **Object Oriented**

  The simulator is fully developed in object-oriented environment. All functions and modules are built in class. Thus, creation or modification of components can be done easily.

University of Malaya

- **Analysis of Simulation result**

  The system provides the log file and meter function. The log file function enable user to analyze the performance of the developed network topology after the simulation while the meter function plots the simulation result to graph during simulation running. These functions enable user to performed further analysis.

- **Flexible traffic conditioner selection**

  The developed simulator enable user to choose either using traffic conditioner (either Leaky Bucket or Token Bucket) or without traffic conditioner to simulate the network topology. Besides that, the simulator also enables the user to choose either global policing or self policing for using the traffic conditioner during simulation. Thus, user can compare the performance between the each policing technique for different profiles.

8.4 Simulator Limitation

The simulator is implemented with a limited set of network component type. These component types are not a complete set of existing component according to NIST Network Simulator standard. The simulation run time is slow because it uses a lots of CPU memory resource. Besides that, Java Virtual Machine is needed in order to run the simulator.

University of Malaya

8.5 Future Enhancement

Although the simulator is powerful and good, but there are some future enhancement suggested as below:

- The simulator graphical user interface is user friendly but the procedure to use every function is not state or not so clear. Therefore, I suggest making a user manual or help file for every function.
- Apart from that, the simulator also having a poor animation or graphics compare to other simulator in the market such as NS 2. These animation or graphics will attract user to use and the procedure to simulate the network is clearer.

8.6 Summary

This project managed to achieve the overall project objectives and requirements as a network simulator system as determined during the system analysis. The simulator testing phase has proved that the project is implemented successfully. This project has taken quite a long time to bring it a success. The experiences gained are memorable and meaningful.

University of Malaya

REFERENCES

1) Forouzan,Behrouz A., 2001.*Data Communications and Networking*. 2nd ed. United States: McGraw-Hill

2) Deitel, H.M and Deitel, P.J., 2003. Java How To Program. 5th edition. New Jersey, United States: Prentice Hall.

3) (Ion 2003) Ion Stoica, Token Bucket Example. University Carnegie Mellon. Available at: www.cs.cmu.edu/~hzhang/15-744/token_bucket.pdf

4) (Ion 2003) Ion Stoica, Differentiated Services (February 25, 2003). Available at: www-inst.eecs.berkeley.edu/~cs268/ sp03/notes/Lecture11.pdf

5) (Juha 2002) Lecture 9 (TDTS41 Computer Networks): Quality of Serviceby Juha Takkinen, IDA/IISLAB, Linköpings universitet (2002-10-07). Available at: www.ida.liu.se/~TDTS41/2002/Lectures/ tdts41-09-qos-6up.pdf

6) (Chimento 2003) Dr. Chimento, The Token Bucket (Leaky Bucket) Model. Available at: qbone.internet2.edu/bb/Bucket.doc

7) (Nikos, 2002) Nikos Drakos, CBLU, University of Leeds, Single And Double leaky bucket meter. Available at: http://linux-ip.net/gl/tcng/node54.html and http://linux-ip.net/gl/tcng/node55.html

8) (Cisco 2003) Policing and Shaping Overview. Available at: http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart4/qcpolts.htm#xtocid241931

9) (Jean 2003) Prof. Jean-Yves Le Boudec and Prof. Andrzej Duda, Quality of Service in IP Networks. Available at: http://icawww.epfl.ch

10) ARMBRÜSTER Heinrich and WIMMER Klaus, *Broadband Multimedia Applications using ATM Networks: High Performance Computing, High Capacity Storage, and High-Speed Communication*, IEEE Journal on Selected Areas in Communication, December 1992.

University of Malaya

11) (J.Clasmann, 2003) J.Glasmann, M. Czermin, A. Riedl, Estimation of Token Bucket Parameters for Videoconferencing Systems in Corporate Networks. Available at: www.lkn.ei.tum.de/lkn/mitarbeiter/josef/ public_html/TB-paper-engl.pdf

12) (Rad 1999) GCRA (generic cell rate algorithm). Available at: http://www2.rad.com/networks/1999/atm/home.htm

13) (Sin Wai Kit, 2001). Sin Wai Kit 2001, ATM NETWORK SIMULATOR with emphasis on Congestion Control. Thesis (PhD). University of Malaya.

14) (WongChee Sum, 2001), ATM Network Simulation with emphasis on Schedule Poilicies in Traffic Shaping. Thesis (PhD). University of Malaya.

University of Malaya

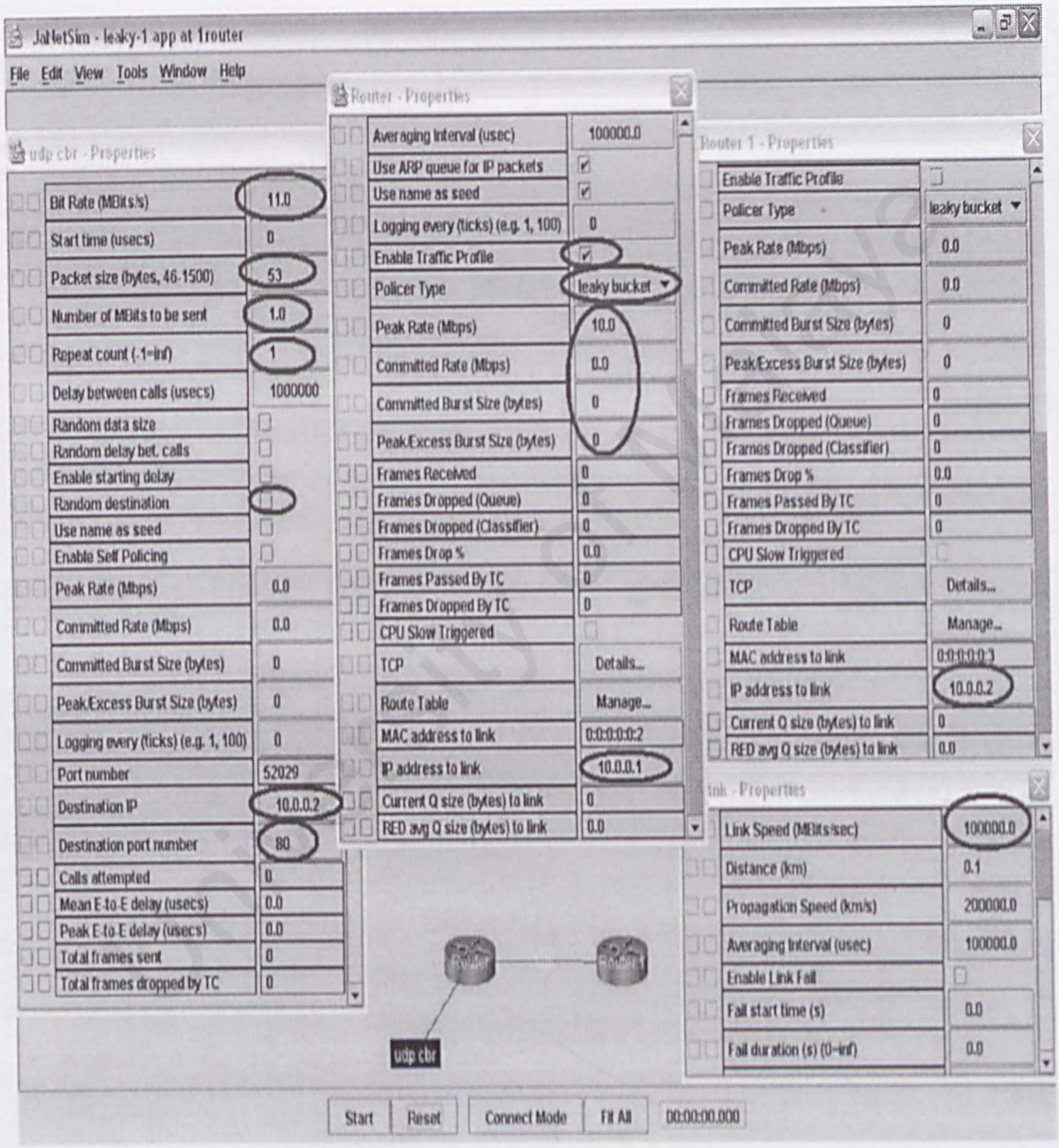
APPENDIX

The following pages shown all the special test case which had been done:

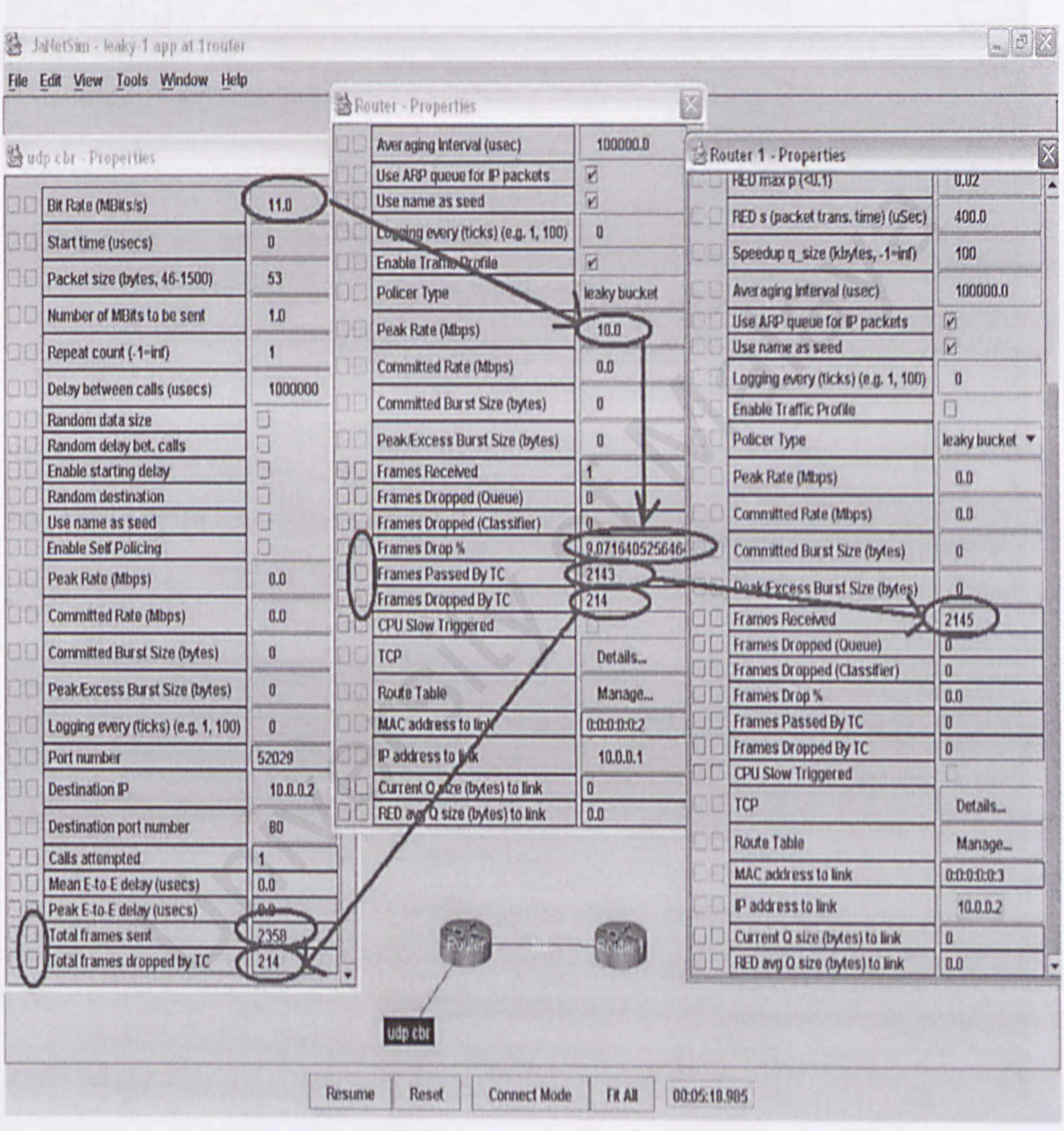
1. Basic setting for the test case
2. Global testing for one Application at one Router
3. Self testing for one Application at one Router
4. Self testing for two Applications at one Router
5. Global testing for two Applications at one Router
6. Self testing for two Applications at two Routers
7. Double Leaky Bucket
8. Controlling Committed Burst Size for Double Leaky Bucket
9. Token Bucket
10. Controlling Committed Burst Size for Token Bucket
11. Using both Leaky Bucket and Token Bucket

University of Malaya

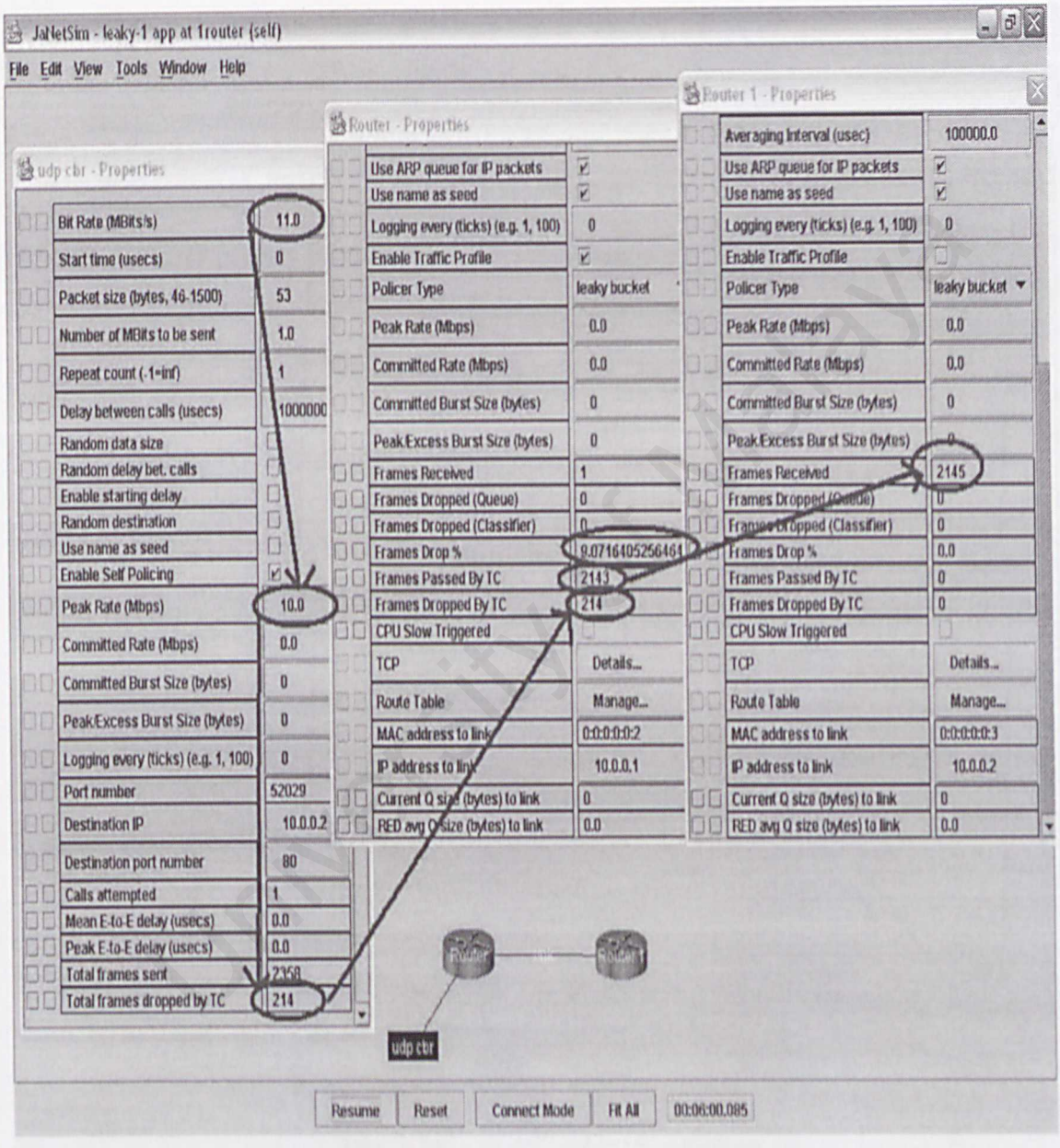
Basic setting for the test case



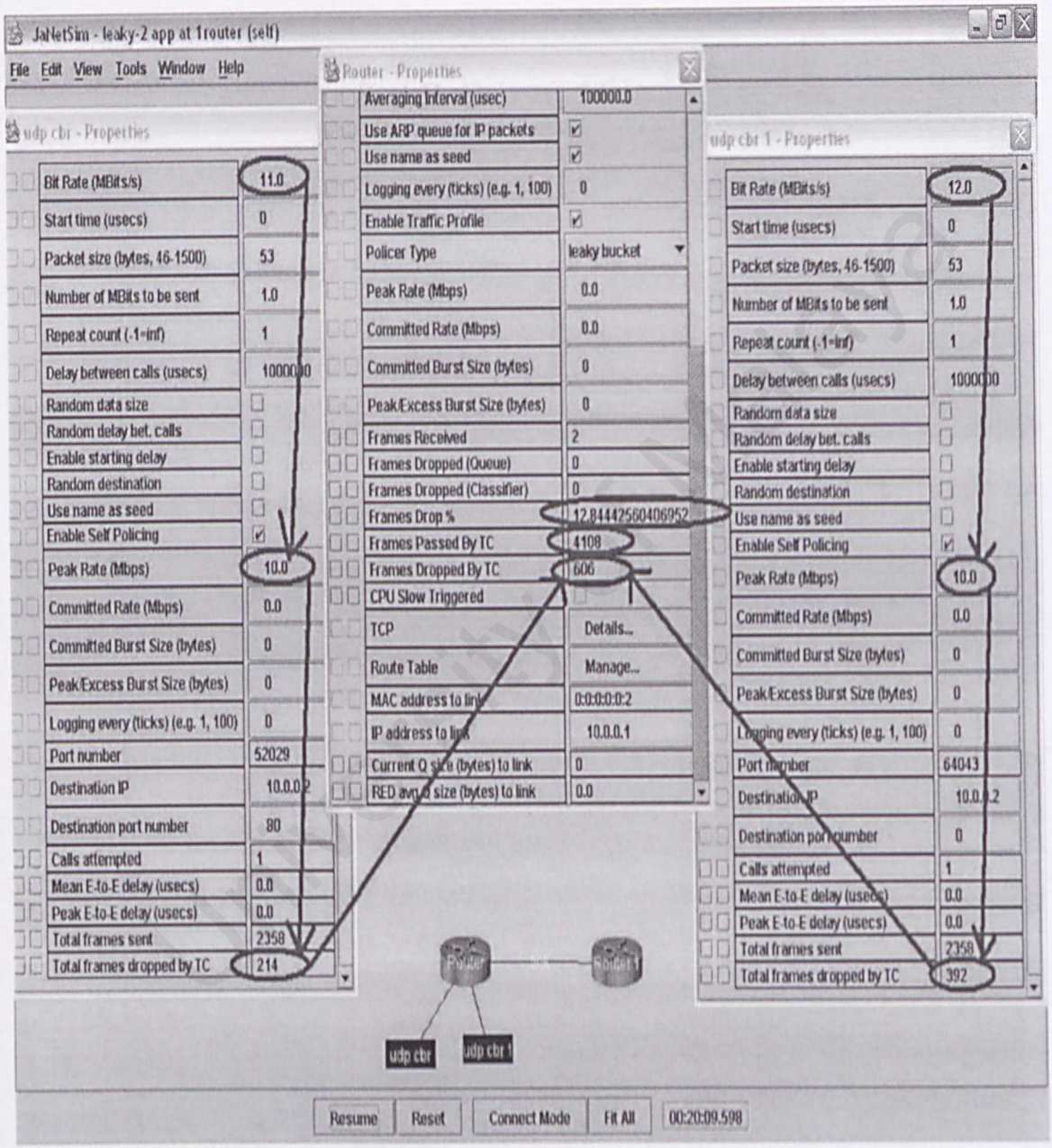
Global testing for one Application at one Router



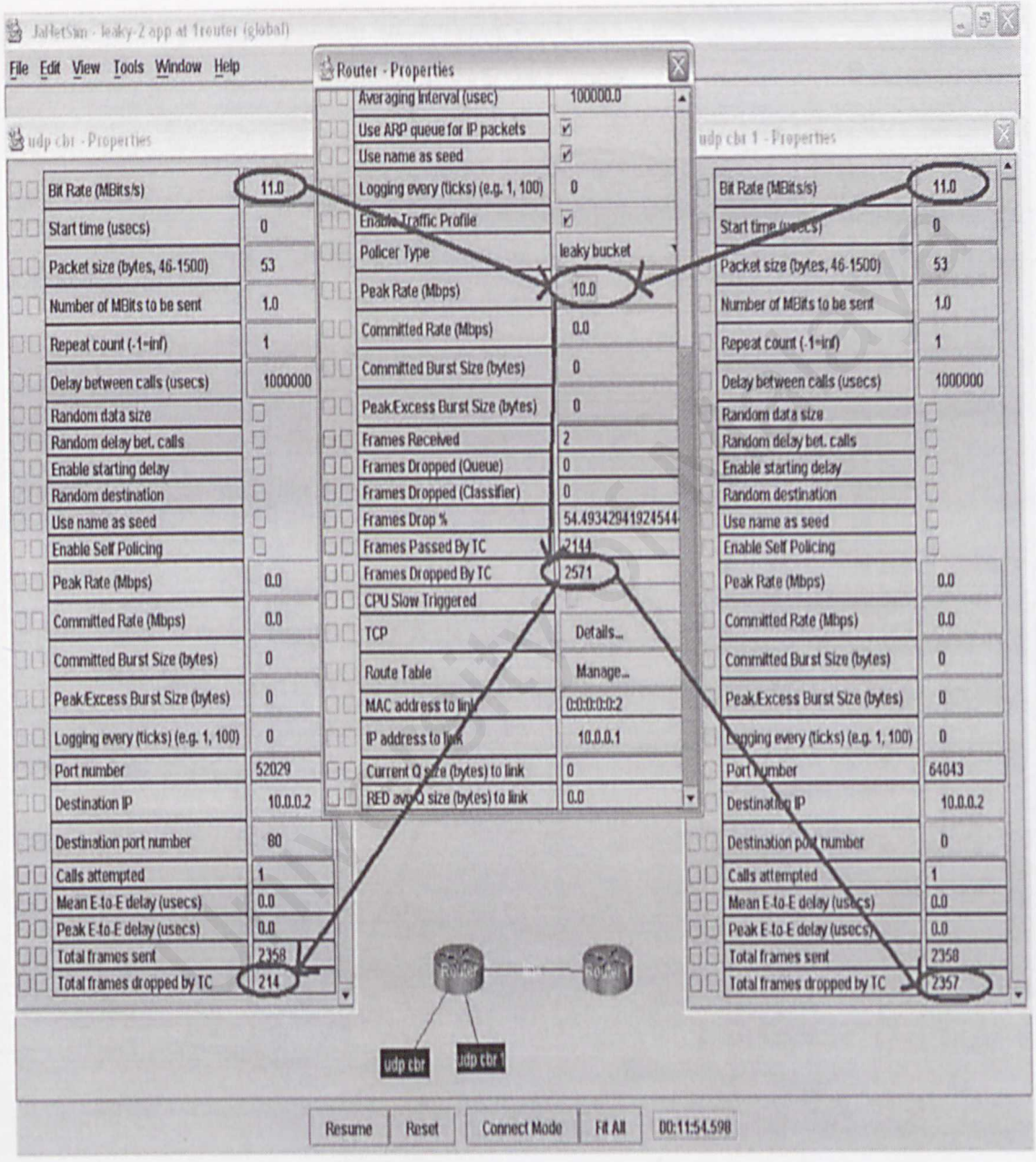
Self testing for one Application at one Router



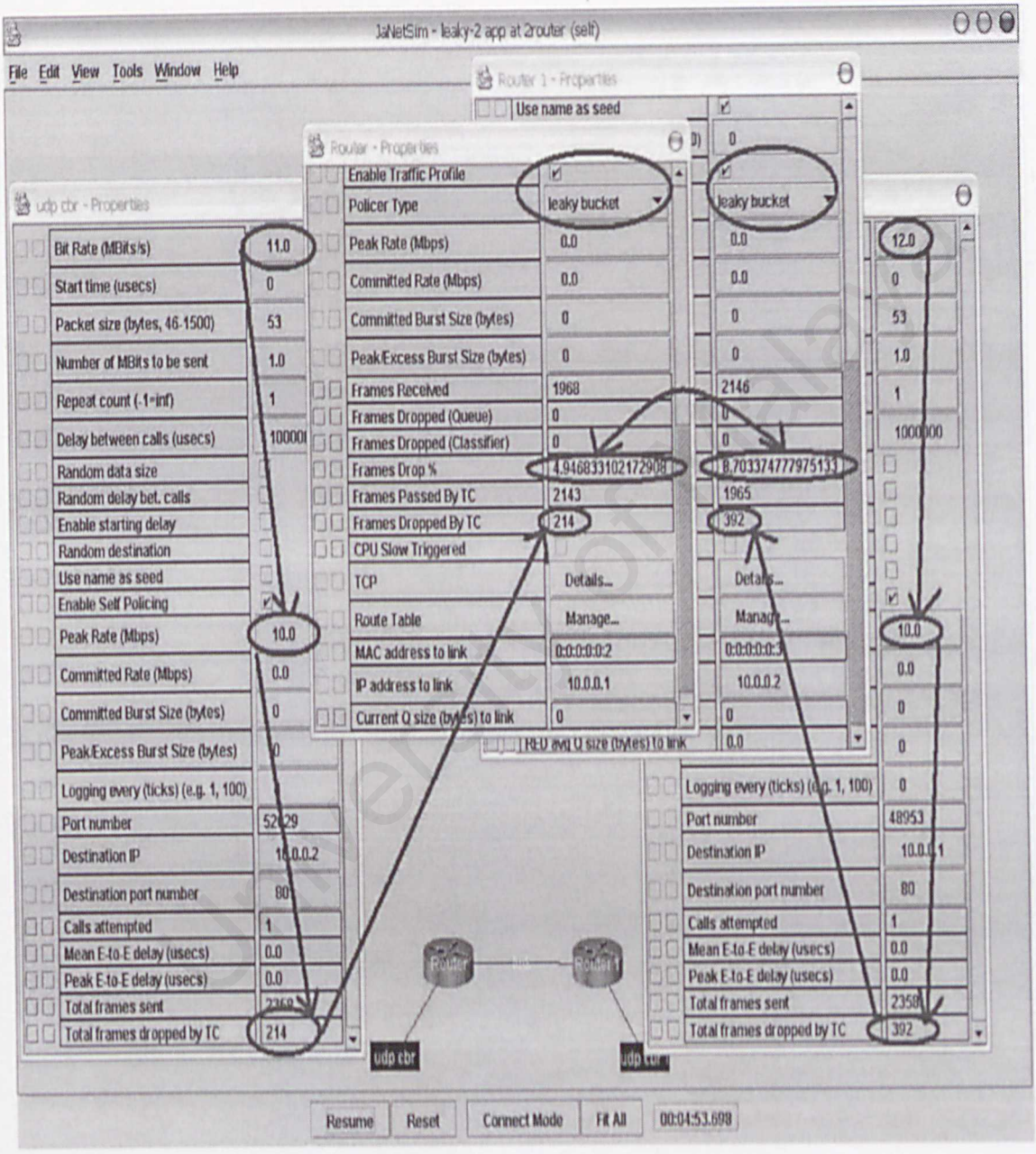
Self testing for two Applications at one Router



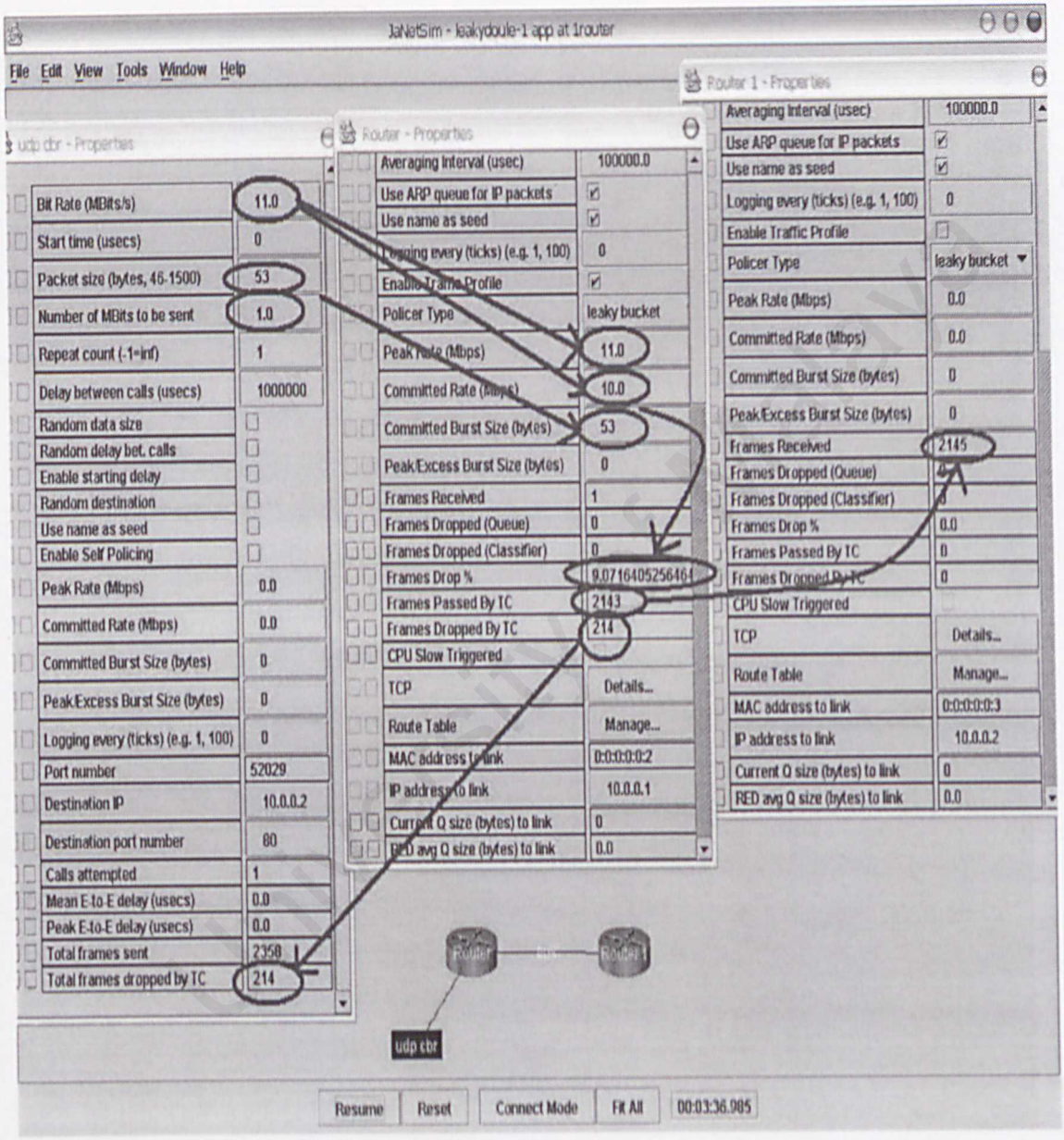
Global testing for two Applications at one Router



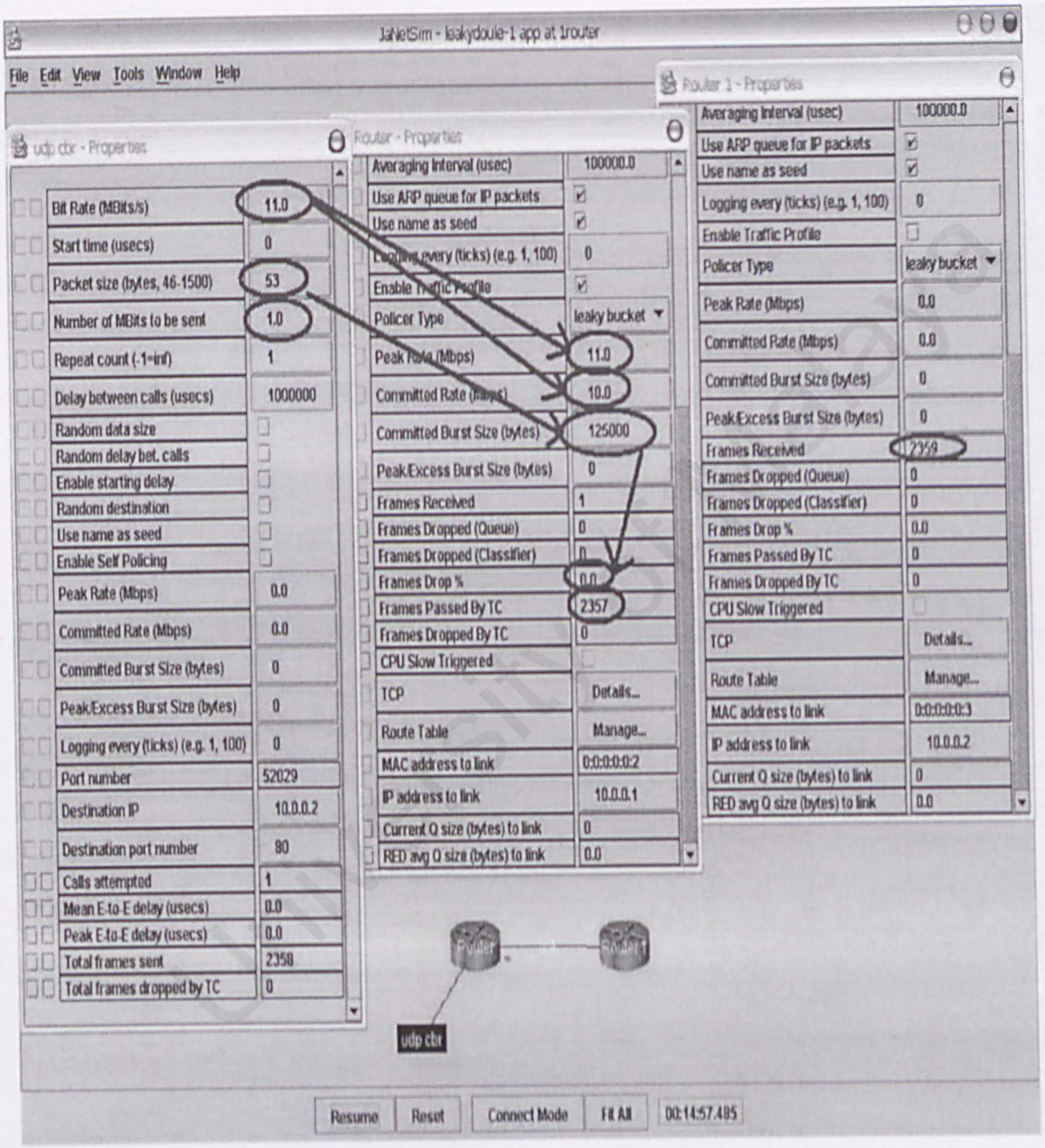
Self testing for two Applications at two Routers



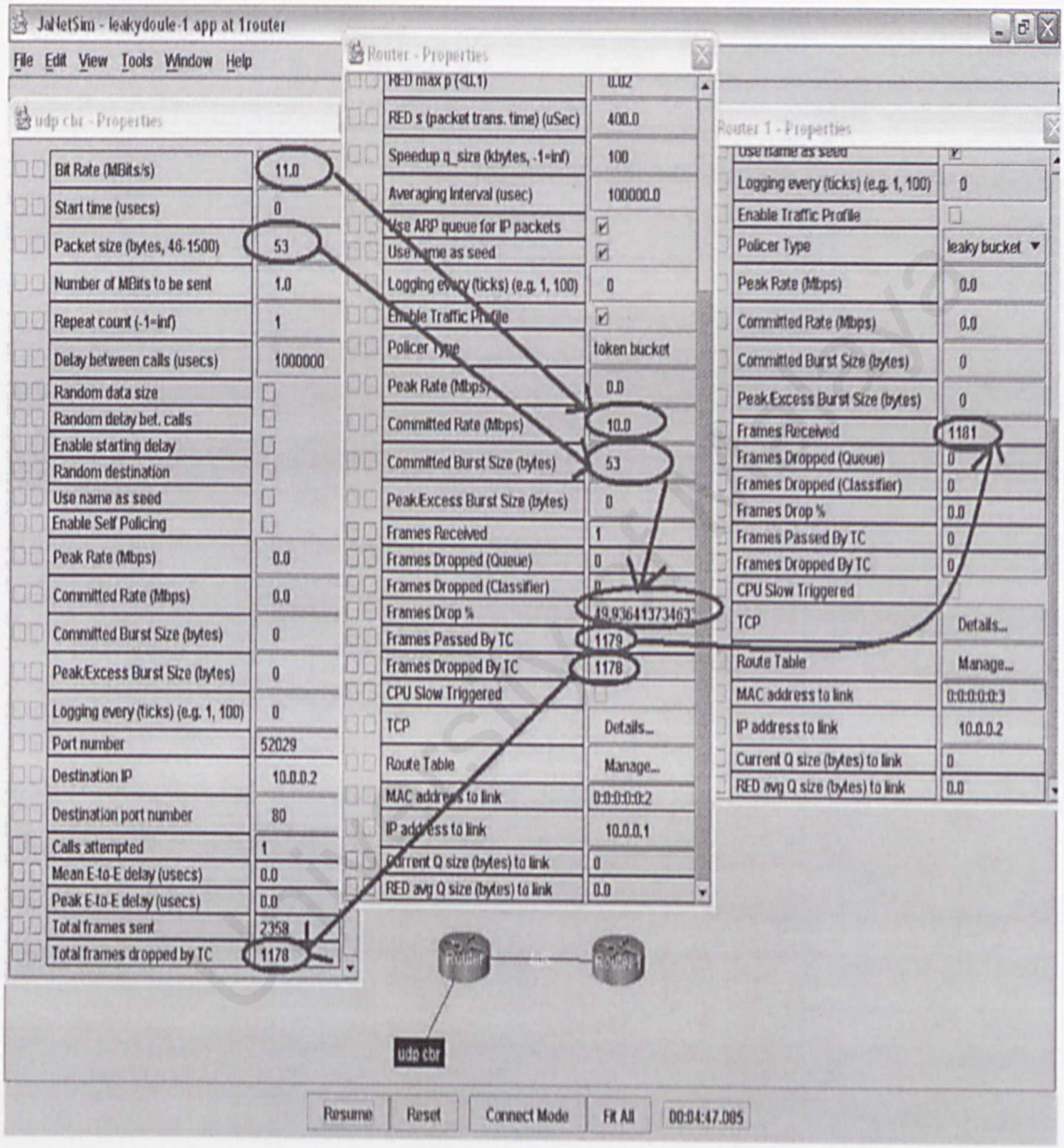
Double Leaky Bucket



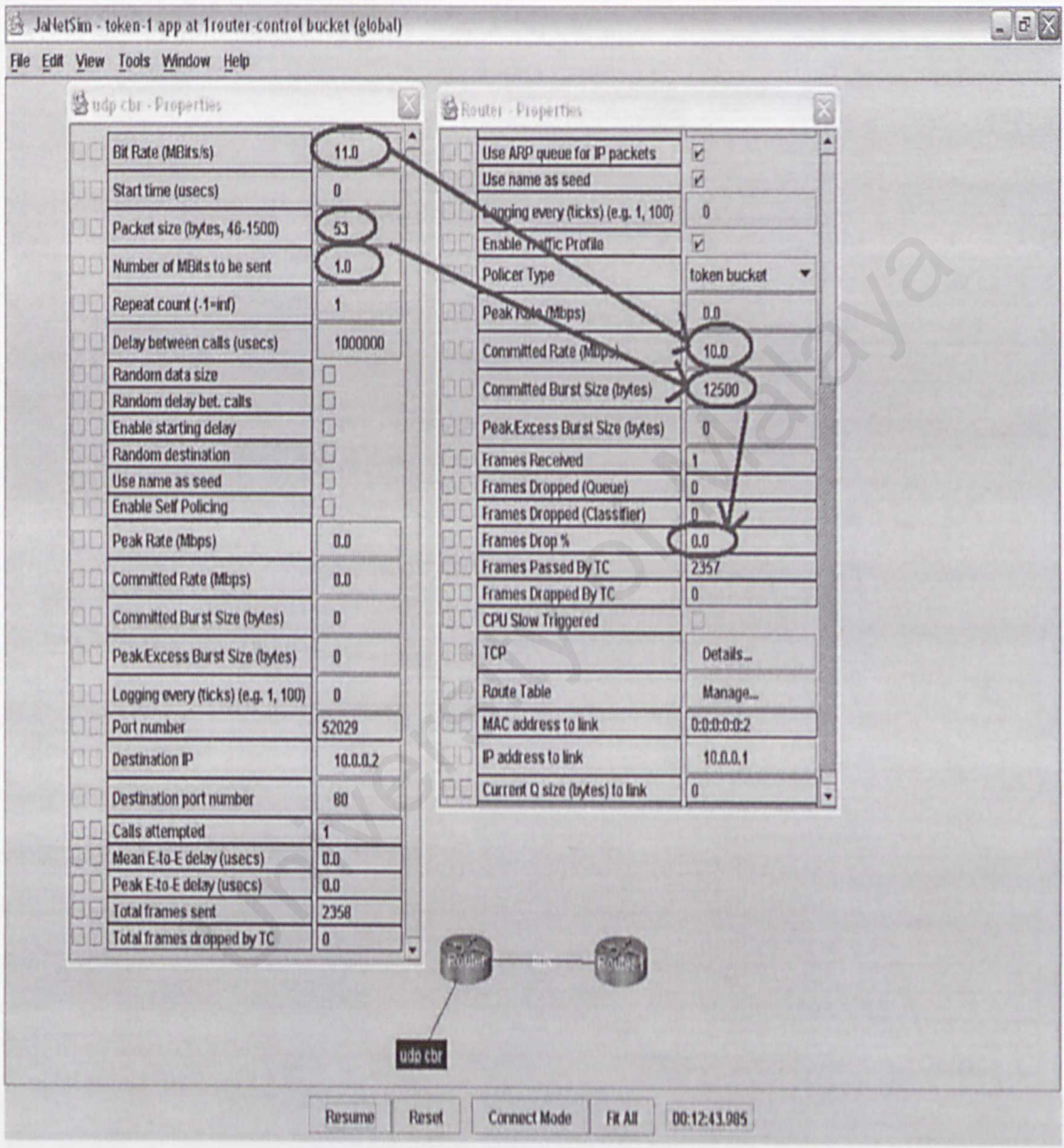
Controlling Committed Burst Size for Double Leaky Bucket



Token Bucket



Controlling Committed Burst Size for Token Bucket



Using both Leaky Bucket and Token Bucket